НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____ Олександр КОВАЛЬ

«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні технології моніторингу довкілля»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Агент системи моніторингу та управління умним домом»

Виконав :
студент IV курсу, групи ТМ-62
Нікітін Олександр Сергійович

Керівник:
к.т.н., доцент
Ковальов Микола Олександрович _____

Рецензент:
к.т.н., доцент
Андреєв Олександр Володимирович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент Олександр НІКІТІН

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший , бакалаврський

зі спеціальності 122 “ Комп’ютерні науки та інформаційні технології”

за спеціалізацією – Інформаційні технології моніторингу довкілля

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр КОВАЛЬ
(підпис)

«___»_____2020р.

ЗАВДАННЯ

на дипломну роботу слухачу

Нікітіну Олександру Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи “ Агент системи моніторингу та управління умним домом“

керівник роботи Ковальов Микола Олександрович , к.т.н., доцент
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від ”25”травня 2020р.**№ 1168-с**

2. Строк подання студентом роботи ’’29’’ травня 2020 р.

3. Вихідні дані до роботи Форма реалізації - Програма з графічним інтерфейсом, реалізована за допомогою Windows Forms API на базі платформи Microsoft .NET розроблена на мові C#. Інтегрована середа розробки – Microsoft Visual Studio. САПР Intel Altera Quartus II, Intel Altera Eclipse.

4.Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити) Аналіз предметної області та існуючих реалізацій систем керування розумним будинком. Програмна реалізація симулятора розумного будинку, апаратна реалізація агента керування «умним домом». Розробка програмного продукту за допомогою вибраних методів та засобів. Створення користувацького графічного інтерфейсу.

5. Перелік ілюстративного матеріалу

«Об’єкт автоматичного керування», «Опис об’єкту автоматизації», «Система клімат-контролю», «Система освітлення», «Апаратна частина для розумного будинку», «Структура процесу розробки», «Розробка апаратної частини для Nios II», «Розробка програмної частини», «Аналіз існуючих рішень автоматизації житлових приміщень», «Етапи розвитку систем автоматизації житлових приміщень», «Структурна схема автоматизації», «Опис існуючих рішень», «Реалізация системи керування “Умним домом”», «Опис програмного продукту “HomeAssistant”», «Алгоритми роботи системи “Умного дому”», «Реалізація алгоритмів освітлення і опалення», «Висновки».

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”13” квітня 2020р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	13.04.2020	
2	Розробка архітектури та загальної структури системи	20.04.2020	
3.	Розробка структур окремих підсистем	27.04.2020	
4.	Програмна реалізація системи	11.05.2020	
5.	Оформлення пояснювальної записки	18.05.2020	
6.	Захист програмного продукту	25.05.2020	
7.	Передзахист	08.06.2020	
8.	Захист	18.06.2020	

Студент _____ Олександр НІКІТІН
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Микола КОВАЛЬОВ
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

до бакалаврської дипломної роботи Нікітіна Олександра Сергійовича
на тему: «Агент системи моніторингу та
управління умним домом»

Дипломна робота присвячена такому сучасному й актуальному досягненню ІТ-технологій як «розумний будинок», а саме - розробці агенту системи моніторингу та керування «розумним будинком».

Апаратна частина спроектована на базі ПЛІС-плати (програмована логічна інтегральна схема, FPGA) розробника. Її основу складає ПЛІС Intel Altera, в якій реалізовано SOC NIOS II, - софт-процесор. Для проектування використовується САПР Intel Altera Quartus II та SOPC Builder. Розробка програмного забезпечення відбувалося в середовищах Microsoft Visual Studio та Intel Altera Eclipse для цього процесора й присвячується програмна частина.

Запропоновано перспективний підхід до створення таких агентів, що відрізняються оптимальним комплексом параметрів швидкодії, функціональності, масштабування, енергетичних, вартісних, масогабаритних та ін. для систем керування «розумним будинком».

Ключові слова : Розумний будинок, система-на-кристалі(SoC), агент, диспетчер, ПЛІС.

АННОТАЦИЯ

к бакалаврской дипломной работе Никитина Александра Сергеевича

на тему : «Агент системы мониторинга и управления умным домом»

Дипломная работа посвящена такому современному и актуальному достижению IT-технологий как «умный дом», а именно - разработке агента системы мониторинга и управления «умным домом».

Аппаратная часть спроектирована на базе ПЛИС-платы (программируемая логическая интегральная схема, FPGA) разработчика. Ее основу составляет ПЛИС Intel Altera, в которой реализовано SOC NIOS II - софт-процессор. Для проектирования используется САПР Intel Altera Quartus II и SOPC Builder. Разработка программного обеспечения происходило в средах Microsoft Visual Studio и Intel Altera Eclipse для этого процессора и посвящается программная часть.

Предложен перспективный подход к созданию таких агентов, отличающиеся оптимальным комплексом параметров быстродействия, функциональности, масштабирования, энергетических, стоимостных, массогабаритных и др. для систем управления «умным домом».

Ключевые слова: Умный дом, система-на-кристалле (SoC), агент, диспетчер, ПЛИС.

ABSTRACT

to the bachelor diploma work of Nikitin Oleksandr Sergiyovich
on the topic: " Agent of the smart home monitoring and management system "

Thesis is devoted to such a modern and relevant achievement of IT-technologies as "smart home", namely - the development of an agent for monitoring and management of "smart home".

The hardware is designed on the basis of a FPGA board (programmable logic integrated circuit, FPGA) of the developer. It is based on FPGA Intel Altera, which implements SOC NIOS II - a software processor. Intel Altera Quartus II CAD and SOPC Builder are used for design. The software was developed in Microsoft Visual Studio and Intel Altera Eclipse environments for this processor and the software part is dedicated to it.

A promising approach to the creation of such agents, which differ in the optimal set of parameters of speed, functionality, scaling, energy, cost, size, etc., is proposed. for smart home control systems.

Keywords : Smart home, system-on-a-chip (SoC), agent, dispatcher, FPGA.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПЛІС – Програмована логічна інтегральна схема

FPGA - Field-Programmable Gate Array (ПЛІС)

HDL – Мова опису обладнання

SoC – Система на кристалі

MQTT – Телеметричний транспорт у черзі повідомлень

ВСТУП

Актуальність :

Протягом тисячоліть людина намагалася створити максимально комфортні умови для проживання. Але, тільки в 21 столітті, нам вдалося наблизитися до цієї мети, за допомогою нових технологій. Система розумний будинок дозволяє створити новий рівень комфортності та безпеки людського житла. А, крім того, така система дозволяє значно економити енергетичні ресурси.

«Розумний будинок» – це високотехнологічна система, яка може об’єднати всі комунікації вашого дому, і керувати ними одним натисканням кнопки. Освітлення, опалення, сигналізація, відеонагляд – це далеко не всі системи, якими можна керувати з допомогою «розумного будинку».

Наприклад, система керування світлом дає змогу запрограмувати світлові сцени у вашому будинку, чи створити видимість присутності господаря вдома під час його відпочинку в іншій країні. Система керування опаленням легко підтримуватиме задану температуру в цілому приміщенні або в окремих кімнатах, понижуючи її чи піднімаючи у відповідності до заданих параметрів.

Мета ДР :

Розробка агенту моніторингу та управління розумним будинком , який має основне співвідношення основних техніко-економічних параметрів і характеристик. Необхідно оптимальне з точки зору функціональності, масштабування, тимчасових, енергетичних, вартісних, масогабаритних і ін. параметрів програмно-апаратного рішення агента «розумного» будинку. Сучасні програмні і апаратні технології, перш за все, - програмована логіка, дозволяють ефективно вирішити це завдання.

Об’єкт ДР :

Об’єктом розробки виступає агент системи моніторингу та управління «умним домом».

Задачі ДР :

- Обзор літератури
- Запропонування загальної структури системи моніторингу і керування
- Розробка апаратної частини агенту
- Розробка програмного забезпечення агента керування «розумним будинком»
- Конфігурування диспетчера високого рівня на базі HomeAssistant

1.ІНЖЕНЕРНІ СИСТЕМИ. ЖИТЛОВІ ПРИМІЩЕННЯ – ОБ'ЄКТ АВТОМАТИЧНОГО КЕРУВАННЯ

1.1 Опис об'єкта автоматизації

Об'єктом автоматизації виступає житлове приміщення в багатоповерховому панельному будинку нестандартного планування. Схема квартири представлена на (рис 1.1). На об'єкті вже присутні певні інжерені системи:

- опалення;
- вентиляція;
- освітлення.

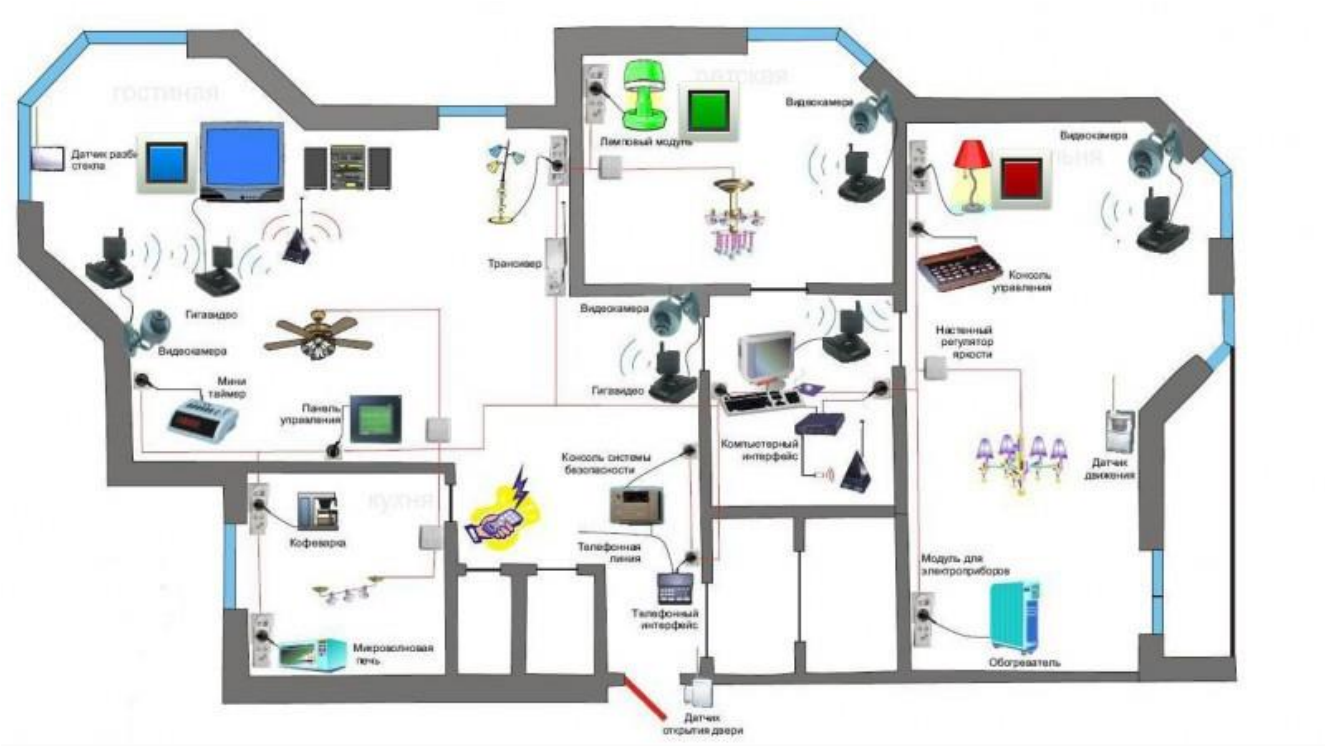

Рисунок 1.1 – Загальна схема автоматично керованого об'єкта

Під час побудови використовувалися сучасні технології утеплення та енергозбереження, в результаті, даної квартирі притаманні певні особливості. Для освітлення використовується набір RGB світлодіодних стрічок, а також наявні стельові світильники.

Система теплої підлоги, а також вмонтовані електричні конвектора, забезпечують вкрай надійне опалення квартири в потрібні терміни. Управління цим обладнанням здійснюється з будь-якої точки квартири, а відсутність гарячого водопостачання компенсується установкою додаткового водонагрівача.

У приміщенні практична вентиляція, яка може працювати в двох режимах, активному і пасивному. В активному режимі роботи вентиляторів включені і працюють на повну потужність, при пасивному режимі вони вимкнені.

Проблема цього обладнання вкрай зрозуміла. Так як все обладнання не автоматизоване і працює тільки в ручному режимі, дозріває питання, а саме що такий підхід не дозволяє в повній мірі реалізувати всі можливості, енерговитратності і рівня комфорту в тому числі.

Для вирішення всіх цих проблем, потрібно розробити ефективний агент управління системами розумного будинку, який зможе організувати роботу всіх його окремих частин в єдине ціле, для підвищення безпеки і зручності керування примішенням.

Всі будівельні роботи вже були виконані в даному приміщенні і об'єкт вже використовується, в результаті для реалізації нашої ідеї потрібно встановити наше розроблене обладнання, що не займає багато місця і часу.

1.2 Система клімат-контролю

На даному об'єкті система клімат-контролю складається з декількох частин, а саме - охолодження здійснюється за допомогою LG ELECTRONICS ARTCOOL SLIM (рис 1.2) , опалення з системою автоматизованого теплої підлоги за допомогою ATLANTIC F17 DESIGN 1500W PLUG (рис 1.3), з можливістю зміни температури з пульта термостата.

Рисунок 1.2. – Кондиціонер LG ELECTRONICS ARTCOOL SLIM

Кондиціонер, який використовується на об'єкті, простий у використанні, має достатню потужність для ефективного охолодження приміщення, має великий набір додаткових функцій, відрізняється зниженим енергоспоживанням і не вимагає витратного сервісного обслуговування.

Управління кондиціонером здійснюється за допомогою пульта дистанційного керування.

Рисунок 1.3. — Конвектор ATLANTIC F17 DESIGN 1500W PLUG

Електроконвектор ATLANTIC F17 DESIGN 1500W PLUG комплектується

вбудованим регулятором і датчиком температури, при цьому має стандартну локальну систему автоматичного керування, яка дозволяє підтримувати встановлені значення температури. В рамках розроблюваної програми, в конвектор вбудована система управління, але вона не є функціональною , тому що показання температури, які знімаються з корпусу електроконвектора, відрізняються від температури повітря у будівлі. Відмінними особливостями цього конвектора є вкрай мали витарти електроенергії, а також доступність його у ціною.

Для управління теплою підлогою використовується терморегулятор Caleo 540 з вбудованим датчиком температури (рис. 1.4).

Рисунок 1.4. — терморегулятор Caleo 540R

Цей терморегулятор дозволяє регулювати і керувати температурний стан підлоги. Налаштування параметрів може проводитися як в ручному режимі, так і з використанням пульта дистанційного керування.

Вентиляція приміщення здійснюється шляхом використання настінних клавішних перемикачів. Загальна потужність системи вентиляції становить 600 Вт.

З огляду на вищевикладене, для зниження енергоспоживання і витрат на експлуатацію необхідно розробити агент з алгоритмами управління і мониторингу системою клімат-контролю. Для підвищення загальної точності виробленого

контролю за параметрами житлового приміщення необхідно встановити в кімнатах окремі датчики температури і вологості. Використання вбудованих в пристрої температурних датчиків буде здійснюватися тільки для захисту цих пристроїв від перегріву. Таким чином, для коректного функціонування системи клімат-контролю потрібне створення надійного агенту керівника.

1.3. Система освітлення

Система освітлення приміщення розділена на кілька окремих типів. Виділяється основне освітлення і чергове. Основне освітлення працює завдяки стельовім світильникам, управління якими здійснюється за допомогою настінних перемикачів. А для чергового освітлення виступає набір світлодіодних стрічок, з власними контролерами, які керуються за допомогою пульта дистанційного управління. Чергове освітлення також виступає в якості нічного. Потрібно автоматизувати включення світла в залежності від знаходження в приміщенні людини, також для зниження витрат на енергію потрібно виробляти і автоматичне відключення освітлення, якщо приміщення не використовується. При такому режимі роботи систему необхідно забезпечити показаннями датчиків присутності людей в приміщенні.

Ринок наповнений дорогими рішеннями систем управління житловими приміщеннями, не універсальними, обмеженими функціонально. В даному проекті пропонується використання пристрою, що виконує роль перемикача, через який проводиться підключення необхідного обладнання.

2.АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ АВТОМАТИЗАЦІЇ ЖИТЛОВИХ ПРИМІЩЕНЬ

2.1 Етапи розвитку систем автоматизації житлових приміщень

Перші спроби комп'ютеризувати житло людини і наділити будинок інтелектом були зроблені ще в 60-ті роки XX століття. Це не дивно, адже саме в цю епоху нові комп'ютерні технології починали своє тріумфальний хід по планеті, наприклад, в 1964 році з'явилася універсальна комп'ютерна система IBM System / 360, т.зв. «Мейнфрейм». Вперше над розробками розумного будинку почали працювати американські інженери: грошей у платників податків США було більш ніж достатньо і вони з радістю сприймали будь-які новинки, які робили їх побут ще комфортніше і зручніше. Основна ідея полягала в створенні єдиної інформаційної мережі будинку, яка буде стежити за мікрокліматом приміщень,безпекоюперимет ра і рядом інших показників, спрямованих на комфорт, безпека, енергозбереження господаря і сім'ї. Однак технології того часу не дозволили реалізувати цю ідею: розумний будинок «розумів» занадто повільно, тому його складно було назвати зр учним.

Після закінчення епохи мейнфреймів, почали успішно розвиватися мережі мінікомп’ютерів. На їх зміну прийшли недорогі індивідуальні електронно-обчислювальних машини, що дозволяли досить швидко знаходити рішення на виникають складні професійні проблеми.

Об'єднані в мережі дані електронно-обчислювальні машини почали набувати все більш широке застосування в системах автоматики різного напрямку.

Зрозуміло, в період розвитку технічного прогресу виникало значна кількість серйозних перешкод в області систем автоматичного управління. Здебільшого автоматичні системи були незалежні, а також системи різних виробників, схожі згідно керуючим функцій, були взаємозамінними. Виробники застосовували власні закриті розробки, які не мали на увазі створення можливостей для використання

результатів розробок інших компаній в сфері систем автоматизації. Через те, що розробки представляли собою автоматизації різних виробників на ринку автоматизації житлових приміщень відкрилися можливості для успішного впровадження нових технологій в області автоматизації житлових приміщень.

Результатом комплексного вирішення завдань автоматизації стало поява інтелектуальних будівель (Intelligence Buildings, Smart House), основою яких стали структуровані кабельні мережі (СКМ). В рамках СКМ для потреб телефонних станцій, комп'ютерних мереж, систем безпеки і т. д., з'явилася можливість застосовувати один і той же кабель. потім розвиток отримали системи мультиплексування каналів зв'язку, що дозволяють передавати одночасно різну інформацію по одній кабельній лінії. Активно розвивається інформатика дозволяла удосконалити системи автоматизації інтелектуальних будівель, таким чином, всім стало ясно, що більшість проектів автоматизації будівель, втрачають актуальність до моменту закінчення їх будівництва.

Через те, що даний напрямок мало великий потенціал зростання на ринку і прибутку, на його розвиток були задіяні великі обсяги фінансування. В результаті вдосконалення розробок даного напрямки сформувалося поняття «розумний будинок».

Що таке «розумний дім»? Звичайно, це не самостійне мисляча будівля. «Розумним» називають сучасну будівлю, в якому всі інженерні системи будівлі об'єднані між собою за допомогою високотехнологічних пристроїв, для забезпечення комфортного та безпечного перебування людини в такому будинку. Більшість побутових пристроїв в «розумному будинку» об'єднуються в єдину домашню мережу, яка також має можливість доступу до загальнодоступних мережі. Основною особливістю таких будинків є об'єднання і використання пристроїв різних виробників в рамках єдиної системи. Сигнали від датчиків, встановлених в кожному приміщенні, надходять до центрального комп'ютера, який обробляє отримані сигнали, і, в залежності від поставленої завдання, генерує керуючі команди для пристроїв, які необхідно використовувати.

Разом з тим, алгоритми взаємодії підсистем в будинку повинні бути гнучкими, і повинні легко пристосовуватися під мінливі потреби власника будинку. Необхідно забезпечувати можливість інтеграції систем будинки один з одним при мінімумі витрат, а також автоматизована система управління та її інженерні підсистеми повинні використовувати у своїй роботі блоковий принцип. Що дозволяє кожній підсистемі здійснювати свої функції незалежно від інших, а також налаштовувати і обслуговувати дану підсистему окремо, без впливу на інші підсистеми.

2.2 Структурна схема автоматизації

Пропонується спроектувати систему автоматизації керування «розумним будинком» на основі бездротової передачі даних, проте в деяких ситуаціях будуть застосовуватися і провідні лінії. структурна схема розробляється приведена на (рис 2.1).

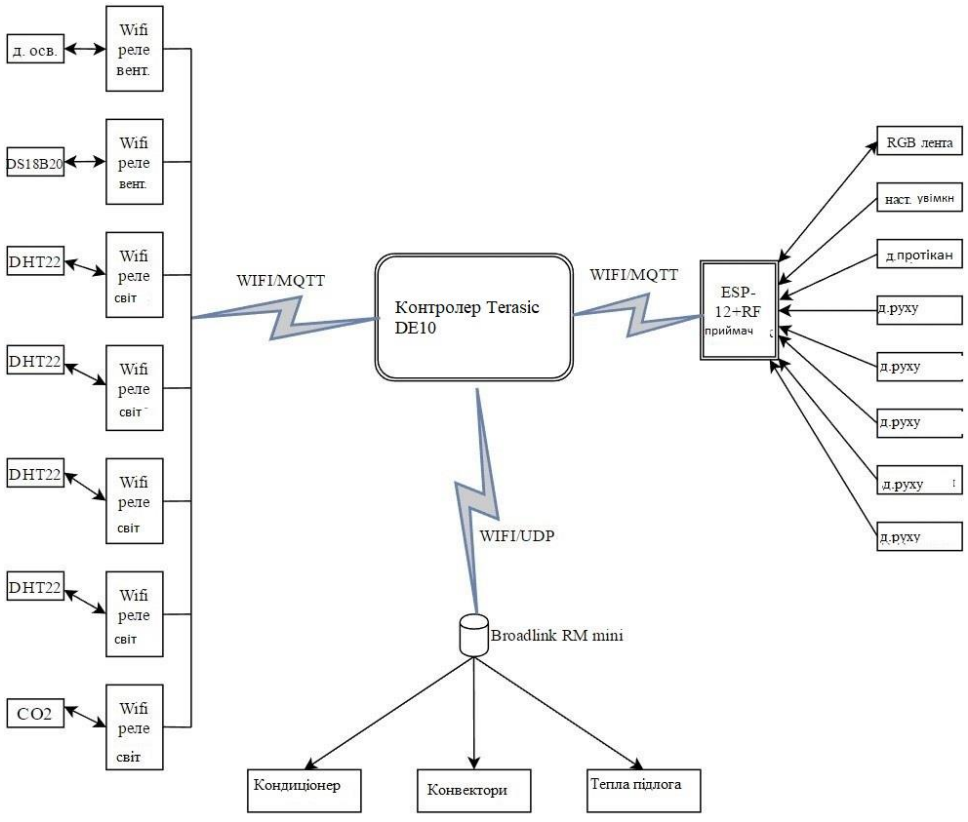

Рисунок 2.1. — Структурна схема системи моніторингу та керування

Об'єкт автоматизації має кілька різних за призначенням і функціоналу кімнат, але схожих з точки зору комфортозабезпечення, найбільш оптимальним рішенням є розміщення головного контролера управління в центрі приміщення, в холі. головний контролер здійснює управління підключеними до нього пристроями, і отримує інформацію з датчиків через бездротову мережу WIFI по протоколу MQTT. Такий варіант розміщення значно знизить вартість системи, тому що це не вимагає прокладки будь-яких проводів, і не доведеться порушувати цілісність ремонту квартири. Так як в квартирі вже є бездротова локальна мережа, то установка додаткового обладнання не потрібно. В свою чергу пропоновані виконавчі пристрої розміщуються в стандартних корпусах побутових розеток, розвідних коробок і приладів, тим багатофункціональним, таким чином, вдасться поєднати в одному блоці кілька функцій, наприклад, в блоці виконавчого пристрою управління освітленням можна вбудувати датчики освітлення і датчики температури і вологості, в блоці виконавчого пристрою управління вентиляцією можна вбудувати вуличний датчик температури.

Для здійснення управління мультимедійною технікою, а також управління кондиціонером, теплою підлогою та конвекторами в автоматичному режимі через ІК діапазон за допомогою розроблюваної системи необхідно підібрати модуль здатний не тільки відтворювати основні команди необхідні для виконання завдання, але також здатний до запису так званому «навчання» від стандартних пультів дистанційного керування.

Різні виробники побутової апаратури застосовують в своїх виробах різні пульти ІК управління. Оскільки пульт повинен спілкуватися тільки з конкретним пристроєм, він формує послідовність даних, унікальну для свого типу обладнання. Передані дані містять крім власне команди управління адресу пристрою, перевірочні дані і іншу сервісну інформацію. Більш того, різні виробники використовують різні способи формування послідовності даних і різні способи передачі логічних станів. найбільш поширені способи кодування бітів інформації - це зміна тривалості паузи між пакетами (метод інтервалів) і кодування поєднанням

станів (біфазної метод). Найбільш поширені формати передачі: RC5 протокол компанії Philips, NEC протокол компанії Philips, JVC, ITT, Mitsubishi, Nokia NRC17, Philips RC6, Phiilps RC-MM, Philips RECS80, RCA Protocol, Samsung, Sharp, Sony SIRC, X-Sat Protocol.

На відміну від пультів управління побутовою електронікою, які передають тільки одну команду, відповідну кнопці, пульти управління кондиціонерами передають при кожному натисканні всю інформацію про параметри, обраних користувачем на екрані пульта, такі як температура, режим охолодження, нагрівання або вентиляції, потужність вентилятора та інші. В результаті, посилка стає досить тривалої. Формати пакетів ІК передачі кондиціонерів: Daikin, Mitsubishi, Samsung.

Обладнання в системі може бути різним, а також може змінюватись його склад при експлуатації, необхідно підібрати універсальний модуль управління цими пристроями.

2.3 Опис існуючих рішень

2.3.1 Clap

CLAP (або Clever Apartment) - розроблена в Україні розумна система керування приміщенням. В 2018 році підписала проект про встановлення у новобудовах систем CLAP. Першим будинком, який був зданий в експлуатацію з системою «розумнийдім» від CLAP був ЖК «Блакитний парус».

Економія. Система допомагає помітно заощадити на комунальних платежах. Якщо джерел тепла в квартирі кілька, CLAP сам вибирає з них те, яке дозволить отримати потрібну температуру з найменшими витратами. Якщо в будинку є знижки на енергоресурси в певні години, система розумного дому врахує і це. Також ви можете встановити оптимальний сценарій енергоспоживання.

Безпека. У разі незаконного проникнення CLAP заблокує двері, включить сигналізацію і викличе охорону. Розумний дім повідомить про займання в квартирі і запустить пожежну сигналізацію, а при затопленні автоматично вимкне воду.

Сервіс та комфорт. Ви можете регулювати температуру в приміщенні, включити обігрів підлоги, налаштувати сценарій опалення під конкретну ситуацію, контролювати витрати на комунальні послуги.

Рисунок 2.2 — Система умного будинку CLAP

2.3.2 Ecoisme

Ця система створена для контролювання витрат електроенергії в побутових та офісних приміщення. Система складається із Центрального модуля, периферії та дозволяє керувати витратами електроенергії та дозволяє встановлювати найбільш поширеніші режими використання електроенергії протягом дня. Є можливість керування сонячними панелями.

Контролер для смарт-будинку складається з датчиків, які збирають дані споживання енергії всіма домашніми приладами. Отримана таким чином статистика підказує господареві, як економити ресурси.

На сьогодні розроблено 2 види контролерів Ecoisme: для будинків, в яких є

фотоелектричні панелі, і для тих, де їх немає. Обидва варіанти датчиків розпізнають пристрої та надають інформацію про використання електроенергії. При цьому Ecoisme для фотоелектричних панелей додатково:

• вказує, яка енергія використовується в даний момент: з мережі або від сонячної панелі;

• вказує вартість електроенергії;

• попереджає про закінчення використання електроенергії від сонця.

Ecoisme поєднується майже з усіма видами інверторів напруги.

Рисунок 2.3. — Центральний блок від компанії Ecoisme

2.3.3 uMuni

UMuni–це технічна платформа, яка дозволяє організовувати енергосервісні об’єкти для мереж закладів, торгівельно-розважальних центрів, для приватних підприємств та офісних приміщень. Основним завданням цієї платформи є паспортизація будівель та їх енергоменеджмент, це самий простий і доступний спосіб автоматизації вашого будинку - чи хочете ви контролювати вхідні двері за допомогою uMuni Cam або увімкнути освітлення у вітальні за допомогою uMuni, цей пакет має все необхідне для перетворення ваш будинок в розумний будинок. Найприємніше те, що все в цьому пакеті дуже просто налаштувати і використовувати, і все без проблем працює разом в додатку Wyze. Всі продукти в Starter Pack працюють з Alexa, IFTTT і Google Assistant.

2.2.4 Xiaomi Mi Smart Home

XiaomiMiSmartHome–це комплект, в якому міститься уся необхідна електроніка для організації «розумного будинку». Поставляється в гарній стильній коробочці, в якій міститься блок управління системою, бездротовий комутатор, датчики безпеки(відкриття вікон та дверей, датчики руху), розумна розетка. «Мозком» Xiaomi Smart Home є шлюз або централь, до якої підключаються датчики та інші девайси. Пристрій підтримує протоколи передачі даних ZigBee і Wi-Fi, має вбудовану сирену, виконує функцію нічника з вибором однієї з 16 млн. Квітів, а також дозволяє слухати на смартфоні онлайн-радіо.

У комплекті йдуть 2 бездротових датчика (руху і відкриття дверей або вікна), розумна розетка і кнопка Mijia Wireless Switch. Останню можна використовувати для віддаленого включення / вимикання пристроїв, як дверного дзвінка або ж для передачі різних повідомлень. Кнопка розпізнає поодинокі і подвійні натискання, тривале утримання.

Для взаємодії з сигналізацією необхідно встановити фірмовий додаток Mi Home. У його англомовної версії присутні китайські ієрогліфи, що створює певні незручності у процесі експлуатації. Але в цілому, підтримка бездротового зв'язку і

протоколу P2P, виключає які-небудь труднощі з мережевими настройками і процедурою підключення датчиків.

Рисунок 2.4. Xiaomi Mi Smart Home Security Kit

Ринок наповнений дорогими рішеннями систем управління житловими приміщеннями, не універсальними, обмеженими функціонально. Включають керування багатьма мультимедійними та IT-сервісами, які швидко розвиваються. Таким чином, ми знайшли оптимальні с точки зору функціональності, масштабування, тимчасових, енергетичних, вартісних, масогабаритних та ін. параметри рішень для управління «умним домом». Після всебічного огляду й аналізу існуючих розробок подібних систем прийнято рішення розробляти його як «систему-на-кристалі» (System-on-a-chip, SOC).

3. АПАРАТНА ЧАСТИНА ДЛЯ РОЗУМНОГО БУДИНКУ

Контролер виступає в якості важливого елемента системи управління, приймає на себе одночасно керуючі, комунікаційні функції і настройку зовнішніх пристроїв, що підключаються до системи.

В даний час триває прискорений розвиток цифрових систем. Мікроконтролери, мікропроцесори, мікросхеми програмованої логіки вже використуються повсюдно, а збільшення інтересу до створення і застосування інтелектуальних систем у величезній кількості напрямків і галузей науки і техніки обумовлює зростання ринку цифрових пристроїв.

Програмована логічна інтегральна схема (ПЛІС), обрана в якості апаратної платформи для реалізації процесорного ядра, представляє собою свого роду конструктор для мікроелектроніки. Конфігуруємо програмовану логіку ПЛІС, можна створювати найрізноманітніші електронні схеми - від простих комбінаторних схем, наприклад, суматора або мультіплексора, до складних систем управління роботизованими комплексами, нейросетевих прискорювачів обчислень для пошукових інтернет систем і блоків управління автономними транспортними засобами.

Часто можна зустріти поняття «отладочна плата» (англ. Development board) - платформа, що має в своєму складі одну або кілька ключових мікросхем (ПЛІС, мікроконтролер і інші), що використовується для того, щоб ітеративним шляхом відпрацювати схемотехничні і програмні рішення. На таких платах з FPGA часто відпрацьовується, або «прототіпіруется», архітектура процесорних ядер перед його запуском в серійне виробництво.

Однак на сьогоднішній день має місце набір тенденцій, які полягають в тому, що:

• по-перше, для збільшення продуктивності все частіше потрібно приміняти спеціалізоване (англ. Custom), а не загальне (англ. Generic) рішення. Дана тенденція

проглядається в багатьох сферах. У наш час все частіше можна чути про «індивідуальному підході» і «обліку чиїх би то не було особливостей». У сфері програмування і цифрової схемотехніки можна говорити про створення та використання «спеціалізованого апаратного забезпечення» (англ. Dedicated hardware) для ефективного вирішення конкретного завдання.

Найбільш показовими прикладами можуть стати спеціалізовані інструкції (англ. Custom instructions), покликані збільшувати виробниковість процесорів за рахунок створюваних під конкретну задачу прискорювачів (англ. Accelerators). На момент написання даного посібника корпорація Intel вела роботи зі створення середовища, що спрощує роботу с мікросхемою, яка об'єднує процесори Xeon® і ПЛІС - Intel Acceleration Stack. Дане середовище покликана якраз забезпечити можливість створення апаратних прискорювачів для функцій, що виконуються на процесорі;

• по-друге, спостерігається активне зростання ринку вбудованих систем і при-ложений для них, і все важливішим стає вага такого фактора, як час виходу продукту на ринок (англ. Time to market). У швидко змінюваному світі все важніше буває першим заявити про себе, першим продемонструвати рішення, навіть в недоробленому, попередньому вигляді. Для завдань, що вимагають створення специали зірованних мікросхем (англ. Application-specific standard product, ASIC), час, витрачений на розробку архітектури і топології, виготовлення чіпів, може стати критичним. Можливість швидкого прототипування схеми робить програмовану логіку все більш затребуваною;

• по-третє, має місце взаємопроникнення апаратних і програмних рішень. Процесорні ядра з фіксованою архітектурою об'єднаються з програмованою логікою, як це зробив Intel. Апаратні, ще недавно недоступні для програмованої логіки процесори Arm® Cortex® отримали свою HDL (англ. Hardware description language) версію, тобто тепер доступні для імплементації в ПЛІС. Те ж саме справедливо і для процесорів MIPS.

Програмісти ПЛІС все частіше стикаються з необхідністю роботи з процесорними системами і, як наслідок, важливістю навичок високорівневого

програмування. У свою чергу, програмісти, які пишуть на мовах високого рівня, що бажають збільшити продуктивність своїх алгоритмів, навіть при наявності таких програмних пакетів, як високорівнева синтез (англ. High level synthesis, HLS), повинні розуміти архітектуру і можливості програмованої логіки, розбиратися в мовах опису апаратури.

Тому пропонується реалізація процесорної системи на базі софт-процесора Nios II® з використанням засобів розробки Intel FPGA.

Оскільки це буде процесор з програмним ядром, має сенс поперед визначити його особливості, порівнявши з ASIC-процесором, тобто з процесором з апаратним ядром. Обидва варіанти реалізації процесорів мають свої переваги і недоліки, а вибір на користь того чи іншого рішення робитися виходячи із специфіки поставленого завдання. На програмованої логікою існує можливість як зібрати написане самостійно мовою HDL програмне ядро, так і скористатися готовими програмними ядра-ми.

Виробник мікросхем програмованої логіки Altera, деякий час назад перейшла під всесвітньо відомий бренд Intel, протягом тривалого часу підтримує свій 32-розрядний процесор Nios II, вва-тануч найбільш широко використовуваним процесором з програмним ядром в індустрії ПЛІС. Процесор Nios II має 32-розрядну RISC (англ. Reduced instruction set computer) мікроархітектуру, тобто володіє архітектурою з обмеженим набором команд2.

До його ключових особливостей слід віднести дві повноцінні значною але відрізняються один від одного версії («збирання») - Nios II Fast і Nios II Economy, можливість роботи в широкому діапазоні тактових частот (понад 400 МГц), апаратна (і програмна з боку інтегрованої середовища розробки) підтримка спеціалізованих інструкцій для апаратного прискорення (до 256), підтримка векторного контролера переривань (до 32), можливість під-ключення швидкої пам'яті з ресурсів програмованої логіки і багато дру-гое. Для програмування під Nios II надається Вбудована середу розроблення (англ. Embedded Design Suite®, IDE).

Вона являє собою користувальницький інтерфейс для роботи з кодом і налагодження програми і, крім усього іншого, включає в себе підтримку операційної системи реального часу.

В основі модуля контролю лежить застосування мікроконтролера. Він представляє собою елемент з функціоналом процесора і периферійного пристрою. У нього також можуть включатися блоки ОЗУ і ПЗУ. Можна сказати, що це найпростіша обчислювальна машина, здатна на самостійну роботу за окремими напрямами операцій. На сьогодні більшість процесорів, що випускаються по всьому світу, фактично являє собою мікроконтролер.

Застосування в них єдиною мікросхеми з «достатньою» продуктивністю дозволять істотно скоротити розміри вироби, мінімізувати показники енергоспоживання, а також скоротити вартість виробництва мікроконтролера і пристроїв, до складу яких він входить. В нині найбільш поширеними виступають контролери, створені на базі RISC- архітектури. Застосування схеми з спрощеним набором використовуваних команд гарантує необхідну швидкодію навіть в умовах, коли тактова частота знаходиться на низькому рівні.

Разом з ОЗУ для контролерів може реалізовуватися вбудована пам'ять, що забезпечує пристрою можливість зберігання певного обсягу інформації. Багато моделей контролерів спочатку не передбачають можливостей підключення до них зовнішніх накопичувачів даних, а найдоступніші за ціною моделі мають пам'ять, що допускає одноразову запис. Затребувані контролери такого типу для пристроїв, в яких спочатку не планується проведення робіт по оновленню використовуваних програм. Більш дорогі зразки припускають багаторазову запис інформації на внутрішню пам'ять з енергетичної незалежністю. У більшості випадків для мікроконтролерів застосовується Гарвардська архітектура пам'яті, що є їх головною відмінністю від простих процесорів. В основі подібної схеми лежить незалежне зберігання даних в ОЗУ і ПЗУ.

З урахуванням всіх описаних вище можливостей, мікроконтролер можна визнати в якості оптимального пристрою, відповідного для створення керуючих систем. У той же час, забігаючи трохи наперед, і припускаючи подальше

розширення функціональних можливостей керуючих пристроїв, необхідно передбачити подальше розширення можливостей контролю з метою більш повного задоволення потреб користувача в перспективі. В такому випадку створення головних пристроїв на основі мікропроцесорів з індивідуальної платою виступає в якості недоцільного рішення, так як друкована плата значно скорочує функціональні можливості контролера, і не дозволяє в повній мірі реалізувати портів введення / виводу. Пов'язано це з тим, що друкована плата виступає в якості цільового продукту, створеного під рішення конкретної завдання, і не володіє необхідним потенціалом модернізації для розширення напрямків застосування виробу. Звичайно, в якості варіанту дій можна передбачити внесення змін до використовувану плату в процесі її проектування, але подібні дії обернуться значними фінансовими витратами на розробку і впровадження нововведення, що автоматично збільшить вартість кінцевого продукту, і знизить ступінь його конкурентоспроможності на ринку. На цьому тлі доцільно розглянути можливість переходу в керуючих пристроях від мікроконтролерів до повноцінним апаратно-обчислювальних платформ, що дозволяє на етапі створення не тільки задати певні параметри і алгоритми, необхідні для роботи, але і передбачити можливості безболісної модернізації обладнання в майбутньому з урахуванням нових виникаючих завдань.

Перехід на повсюдну автоматизацію ставить завдання по створенню пристроїв на нових принципах, при цьому сьогодні на ринку представлений широкий вибір апаратно-обчислювальних платформ, володіють необхідним потенціалом. Серед найбільш поширених продуктів можна виділити Parallax Basic Stamp, Phidgets, Espressif та інші. найбільше поширення на ринку мають Raspberry Pi, Arduino, ESP8266. Відмінними рисами даних моделей виступає величезний функціонал, велика кількість доступних програмних продуктів, а також можливість коммутирования з метою розширення наявних технічних можливостей. При всьому при цьому, своїми геометричними розмірами подібні пристрої відповідають банківських картах, що істотно розширює сферу їх застосування.

Будемо реалізовувати процесор на ПЛІС. Оскільки це буде процесор з програмним ядром, має сенс заздалегідь визначити його особливості, порівнювати з ASIC-процесором, тобто з процесором з апаратним ядром. У таблиці 1.1 Представлені особливості обох рішень.

Таблиця 1.1

Порівняльна таблиця, що описує переваги і недоліки двох видів реалізації процесорного ядра Параметр	Процесор з апаратним ядром (Hard-core)	Процесор з програмним ядром (Soft-core)
Максимальна тактова частота роботи	одиниці ГГц	сотні МГц
Місце, займане на кристалі	Менше, ніж в разі програмного рішення, через використання оптимальної кількості і типу логічних елементів	Більше, ніж в разі апаратного рішення, через надмірність програмованої логіки для вирішення специфічного завдання (зокрема імплементації процесорного ядра)
Гнучкість, можливість настройки	З огляду на те, що архітектура і розташування процесорного ядра на кристалі фіксована, можливість настройки обмежена закладеної заздалегідь надмірністю.	Широкі возможности налаштування параметрів під конкретну задачу. Кроме того, з'являється можлівість імплементації кількох процесорних ядер, і їх розташування в тому місці кристала, де це необхідно
Можливість поновлення та повторне використання	У пристроях, де використовуються процесори з апаратними	У пристроях, де використовуються процесори з програмними

	ядрами, оновлення пр оцесора неможливо. З цієї ж причини повторне використанн я пристроїв тва для інших цілей утруднен о. процесор на інший, або відмовитися від процесора на користь системи на програмов аної логікою (HDL). Так само подібні пристрої можуть бути повторно використані для альтернативних завдань за допомогою повної переробки системи.	ядрами, існує можливі сть повністю оновити систему, замінившипр оцесор на інший, або відмовитися від проце сора на користь систе ми на програмованої логікою (HDL). Так само подібні пристрої можуть бути повторно використані для альтернативних завдань за допомогою повної переробки системи.

Використовуємо плату :

Terasic DE10-Standard Development Kit (рис 3.1).

Ціна:	$350
Академічна ціна:	$259

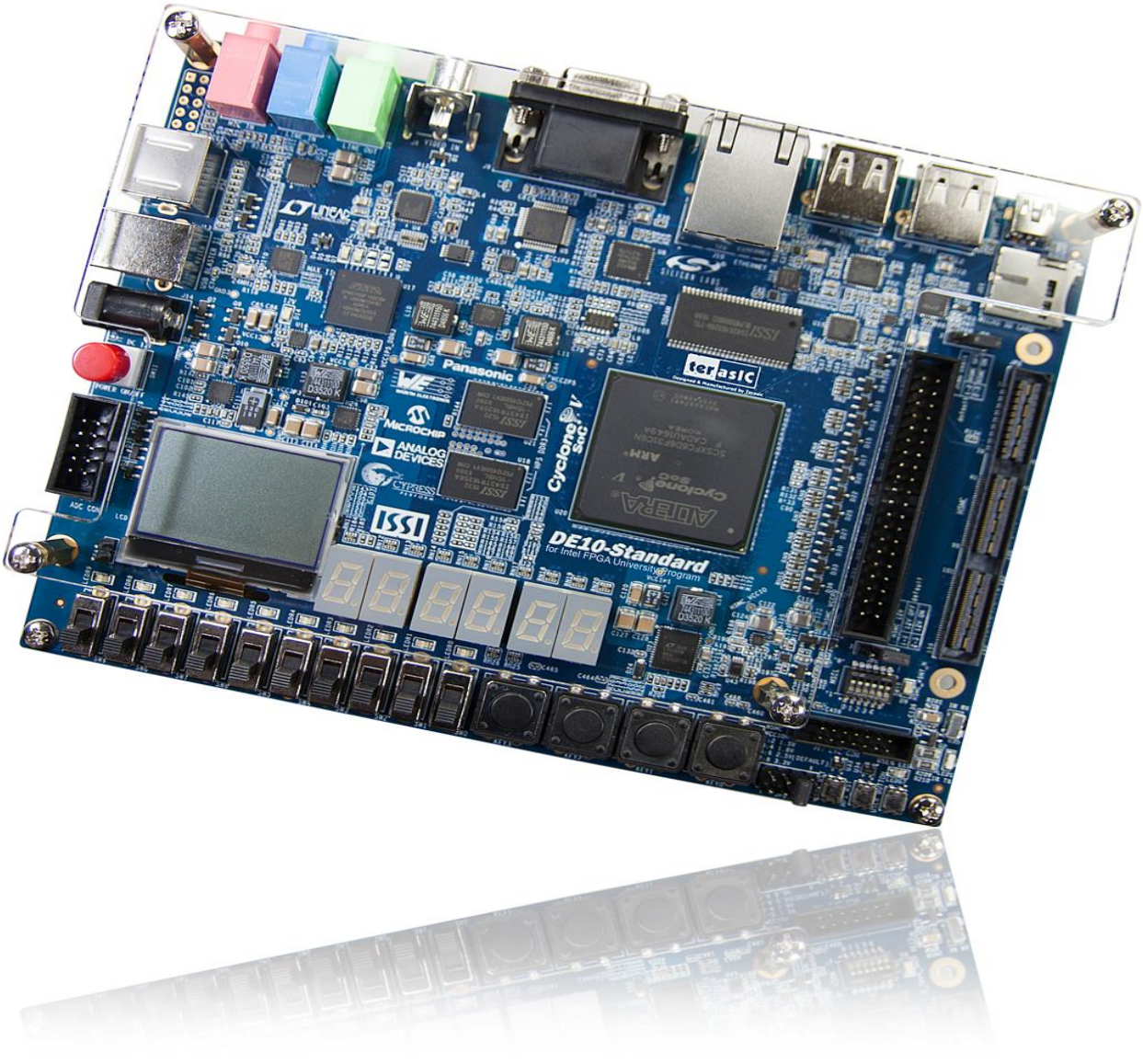

Рисунок 3.1 — Terasic DE10-Standard Development Kit

Порівняння плат :

	 Terasic DE10-Standard	 Terasic DE1-SoC	 Intel Cyclone V SoC Development Board	 Arrow SoCKit (from Terasic)	 Macnica Helio Board	 Xilinx Zedboard	 Digilent Zybo Zynq-7000 ARM/FPGA SoC Trainer Board
FPGA Device	Cyclone V SoC 5CSXFC6D6F31C6N	Cyclone V SoC 5CSEMA5F31C6N	Cyclone V SoC 5CSXFC6D6F31C6N	Cyclone V SoC 5CSXFC6D6F31C6N	Cyclone V 5CSXFC6C6U23C8N	Xilinx® XC7Z0201CLG484C	Xilinx Zynq-7000 (XC7Z010-1CLG400C)
FPGA Logic Elements	110K	85K	110K	110K	110K	85K	28K
Processor	ARM Cortex-A9 Dual-Core	ARM Cortex-A9 Dual-Core	ARM Cortex-A9 Dual-Core	ARM Cortex-A9 Dual-Core		ARM Cortex-A9 Dual-Core	
HPS SDRAM	1GB DDR3	1GB DDR3	1 GB DDR3	1 GB DDR3	1GB DDR3	512MB DDR3	512 MB DDR3
FPGA SDRAM	64MB SDRAM	64MB SDRAM	1 GB DDR3	1 GB DDR3			
Flash	EPCS128		1 Gb QSPI Flash	1 Gb QSPI Flash	1Gb QSPI Flash (Option)	256Mb QSPI Flash	128 Mb QSPI Flash
USB	Host x2	Host x2	OTG x1	OTG x1	OTG x1	OTG x1	OTG x1
Ethernet	1 Gigabit	1 Gigabit	1 Gigabit + 2x10 / 100MB	1 Gigabit	1 Gigabit	1 Gigabit	1 Gigabit
UART	UART to USB	UART to USB	UART to USB	UART to USB	Uart to USB	UART to USB	Uart to USB
PCIe			PCIe x4				
Video In	Composite-in	Composite-in					HDMI Port
Video Out	24-bit VGA DAC	24-bit VGA DAC	SDI	24-bit VGA DAC		VGA / HDMI	16-bit VGA Output port/HDMI
Audio	24-bit Audio Codec	24-bit Codec		24-bit Codec		Audio Codec	Audio CODEC
FPGA Expansion	40-pin GPIO Connector x1 HSMC Connector x1	40-pin GPIO x2	HSMC x1	HSMC x1	HSMC x1	FMC x1 / Prnod x5	12-pin Pmod x4
HPS Expansion	LTC Connector (I2C/SPI)	LTC Connector (I2C/SPI)	LTC Connector (I2C/SPI)	LTC Connector (I2C/SPI)			Pmod x1
User LED/Button/ Switch/7-SEG	11 LEDs / 10 Switches / 5 Buttons / 6 7-SEGs	11 LEDs / 10 Switches / 4 Buttons / 6 7-SEGs	8 LEDs / 6 Switches / 8 Buttons	8 LEDs / 8 Switches / 8 Buttons	4 LEDs/ 8 Switches / 7 Buttons	9 LEDs / 8 Switches / 7 Buttons	5 LEDs/ 4 Switches/ 6 Buttons
LCD	128x64 dosts LCD Module		16x2 Character LCD	128x64 dots LCD Module		128x32 OLED	
ADC	8CH / 12-bit	8CH / 12-bit					4CH/12bit
Sensor	G-Sensor	G-Sensor		G-Sensor / Temperature Sensor			Temperature Sensor
Other Interface	PS/2 & IR Emitter / Receiver	PS/2 & IR Emitter / Receiver	CAN				
Price	Academic: $259 Industrial : $350	Academic: $175 Industrial : $249	$1.795	$350	$249	$495	$199
Summary	Recommended	Recommended					

Рисунок 3.2 — Порівняння плат

Система на основі процесора Nios II складається з двох окремих частин - апаратної і програмної. Вбудований в Quartus інструмент Platform Designer® використовується для реалізації апаратної частини: настройка конфігурації процесора і периферії. Платформа Nios II EDS використовується для роботи з програмною частиною проекту. У цьому розділі ми зберемо базову систему для того, щоб продемонструвати процес розробки апаратної частини і особливості роботи з програмною частиною проекту: розглянемо послідовність і специфіку розробки додатків для збирання апаратної системи.

3.1 СТРУКТУРА ПРОЦЕСА РОЗРОБКИ

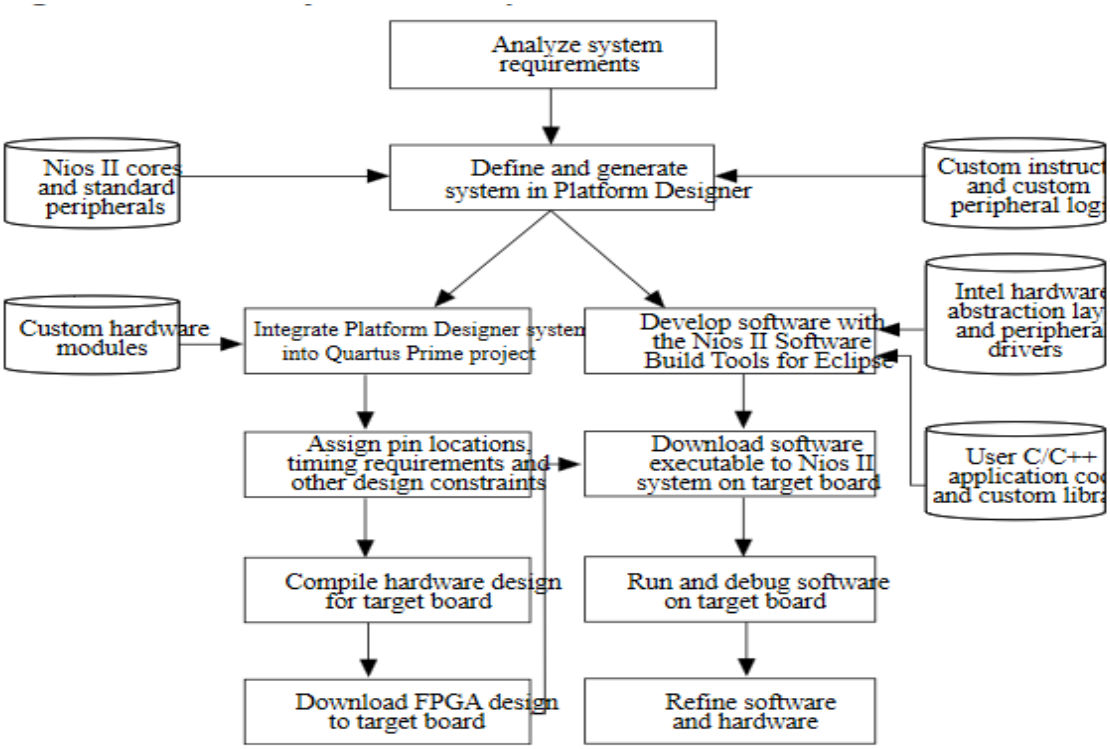

Рисунок 3.3 — Схема процеса розробки на Nios II

3.1.1 Розробка апаратної частини

При зростанні складності розроблюваних систем значно збільшується час, необхідний для реалізації необхідного функціоналу (ліва частина схеми). Одним з рішень такої проблеми є використання готових модулів в проекті. Подібні модулі можуть бути розроблені як самим розробником, так і сторонніми програмістами, і часто оформляються у вигляді інтелектуального продукту, виконаного за певними стандартами, тому їх прийнято називати IP-ядрами (Intellectual Property cores), тобто об'єктами інтелектуальної власності. Такі ядра можуть бути безкоштовними, можуть бути спеціалізованими під певного виробника, а можуть коштувати десятки тисяч доларів США.

Intel FPGA надає великий вибір таких модулів, частина з яких є безкоштовними і доступними всім користувачам. Для розробки систем на основі готових модулів в Quartus доступний додатковий інструмент - Platform Designer.

Даний інструмент дозволяє значно заощадити час, автоматизуючи більшу частину генерації логіки межсоединений. За результатами складання системи Platform Designer перевіряє збірку на помилки і генерує HDLкоди для подальшої компіляції та файли з розширенням * .qsys і * .sopcinfo, що зберігають в собі інформацію про конфігурацію системи. Ці файли в подальшому використовуються для компіляції та генерації файлів для програмної частини проекту.

3.1.2 Постановка завдання для першого проекту

Процес розробки системи на основі процесора Nios II складається з декількох етапів. Для ілюстрації процесу розробки системи ми зберемо невелику систему управління світлодіодами. Смуга з десяти світлодіодів буде імітувати візуалізацію процесу завантаження - світлодіоди послідовно будуть загорятися один за іншим, поки всі вони не почнуть світитися, після чого вони все згаснуть, і так далі. Десять перемикачів на платі будуть регулювати швидкість, з якою світлодіоди будуть запалюватися. Для подібної простий системи ми скористаємося мінімальною конфігурацією процесора Nios II - Economy.

Головним завданням цього проекту є ознайомлення з програмним забезпеченням, необхідним для побудови подібних і більш складних систем на базі Nios II. Основними кроками в розробці проекту будуть:

1.Створення апаратного проекту в Quartus;

1.1.Збірка системи на основі Nios II і генерація HDL-коду за допомогою Platform Designer;

1.2.Додавання системи в файл верхнього рівня проекту, підключення системи до інших модулів проекту та портів ПЛІС;

1.3.Компіляція проекту і конфігурація ПЛІС, ознаменовує собою завершення створення апаратної частини проекту;

2.Генерація бібліотеки BSP;

3.Написання коду програми на мові Сі;

4.Складання програмного проекту і запуск його на платі.

3.2 Розробка апаратної частини проекту Nios II

Для реалізації апаратної частини проекту необхідно виконати наступні дії :

- Створити новий проект в середовищі розробки Quartus;
- Зібрати просту систему на базі процесора Nios II з використанням інструменту Platform Designer:

o Додати і конфігурувати модулі периферії, пам'яті, ідентифікатора системи і процесора;

o Вибрати коректні підключення в матриці межсоединений і експортувати з системи необхідні порти;

o Визначити вектора скидання і винятків;

o Визначити базові адреси; o Згенерувати HDL-файли і bsf-файл;

- Створити файл верхнього рівня, в якому буде присутній зібрана система, і підключити истеми до портів ПЛІС;
- Налаштувати периферію за допомогою інструменту Pin Planner;
- Скомпілювати проект, конфігурувати (на сленгу - «прошити») ПЛІС.

3.2.1 Створення проекту в Quartus

Щоб створити збірку системи на базі Nios II, необхідно створити проект. Для цього запустіть середу Quartus і створіть порожній проект у верхній лівій панелі Файл> Майстер нових проектів. У вікні Введення з'явиться опис созание проекту (рис 3.4). Натисніть Далі>.

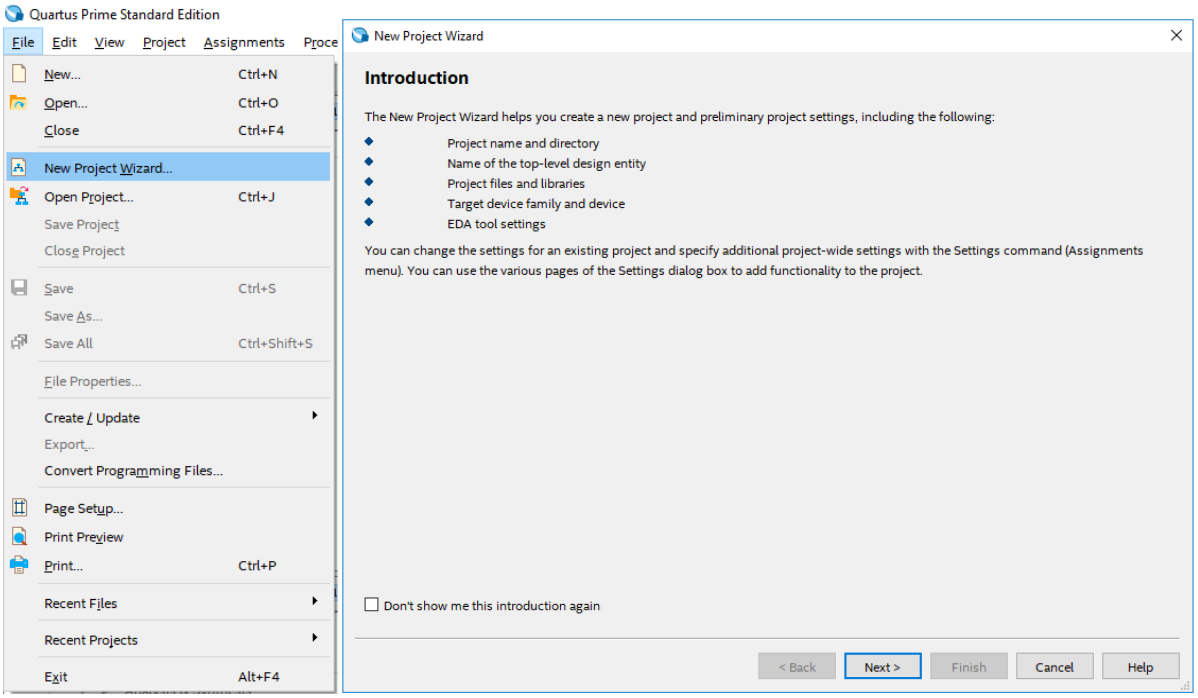

Рисунок 3.4 — Перші кроки по створенню проекту в середовищі Quartus.

Наступним кроком Directory, Name, Top-level Entity необхідно вказати директорію, в якій буде зберігатися проект і всі його файли, і визначити назву проекту (рис 3.5). За замовчуванням вважається, що файл верхнього рівня має ту ж назву, що і сам проект.

Даний проект реалізується з нуля, тому в наступному кроці Project type вибирається пункт Empty project. З цієї ж причини крок Add files пропускається. В наступному кроці, Family, Device & Board Settings, необхідно визначити, для

якого саме сімейства ПЛІС розробляється проект (рис 3.6). У цьому проекті використовується плата DE10-Standard від тайванського виробника Terasic, на якій встановлений кристал з артикулом (part number) 5CSXFC6D6F31C6N. Даний чіп відноситься до сімейства Cyclone V, це необхідно вказати в випадаючому списку пункту Family. У списку доступних пристроїв необхідно поставте галочку напроти цього кристал.

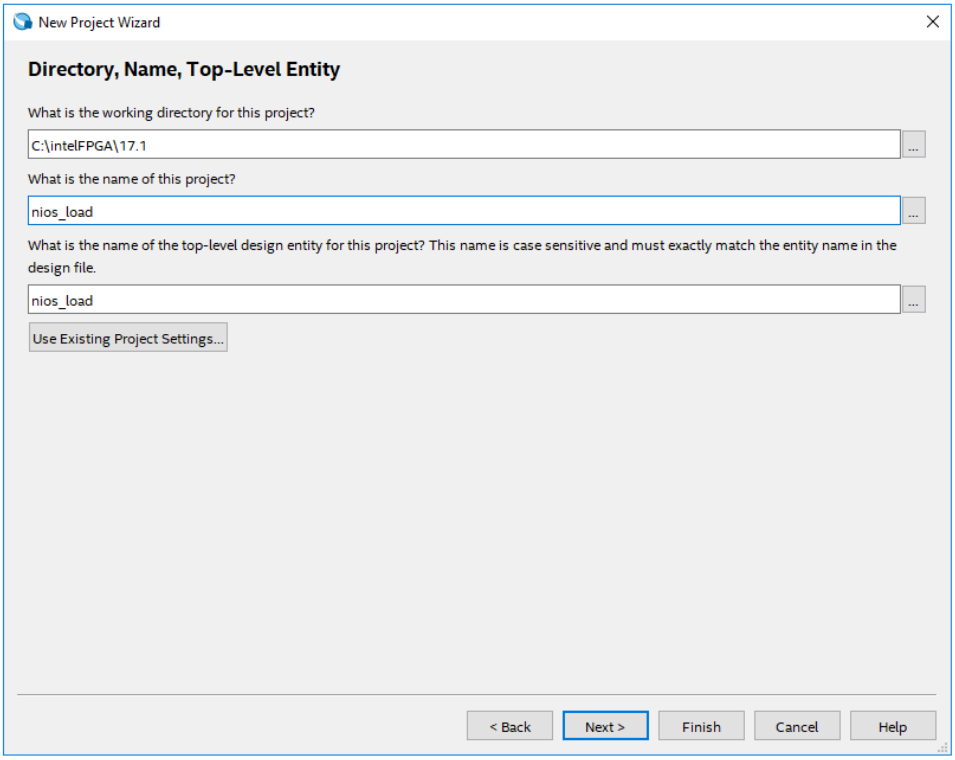

Рисунок 3.5 — Вікно вибору директорії і назви для проекту і файлу верхнього рівня

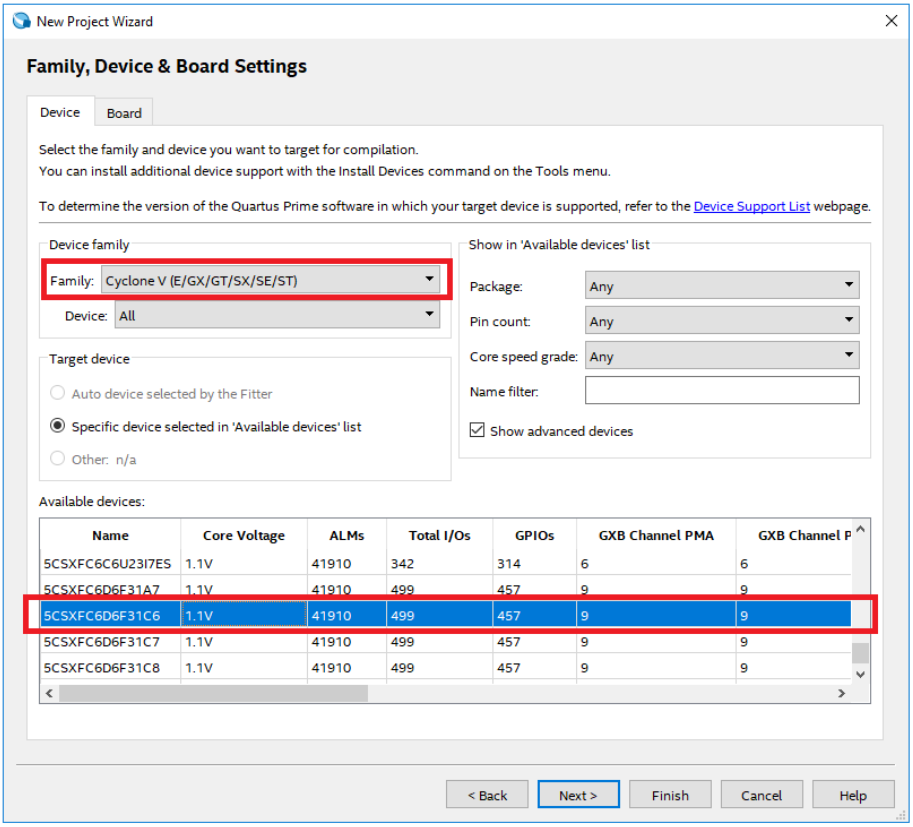

Рисунок 3.6 — Вибір сімейства (family) і пристрої (device), для якого створюється проект

Цих налаштувань досить для поточного проекту, тому можна натиснути кнопку Finish. Тепер можна приступити до складання системи на основі Nios II в середовищі Platform Designer.

3.2.2 Збірка процесорної системи nios_load1 в Platform Designer

Запустити інструмент Platform Designer можна як натиснувши іконку на панелі інструментів, так і через меню Tools> Platform Designeer. Після ініціалізації відкриється порожня система, в якій є тільки джерело тактового сигналу (рис 3.7). У лівій частині екрана знаходиться набір тематично підібраних бібліотек, IP Catalog, елементи якого ми будемо додавати в систему. На вкладці System Contents відображаються всі додані в яку збирають систему блоки, їх межсоединения і різна інформація, включаючи базову адресу, пріоритети переривань, експортовані порти тощо. Приступимо до збірки системи.

Додавання процесора Nios II Gen2 :

У вікні IP Catalog вибираємо категорії Processors and Peripherals> Embedded Processors (або знаходимо за допомогою пошуку) і вибираємо компонент Nios II Processor, натискаємо Add, щоб додати його в систему, з'являється вікно настройки процесора (рис 3.8).

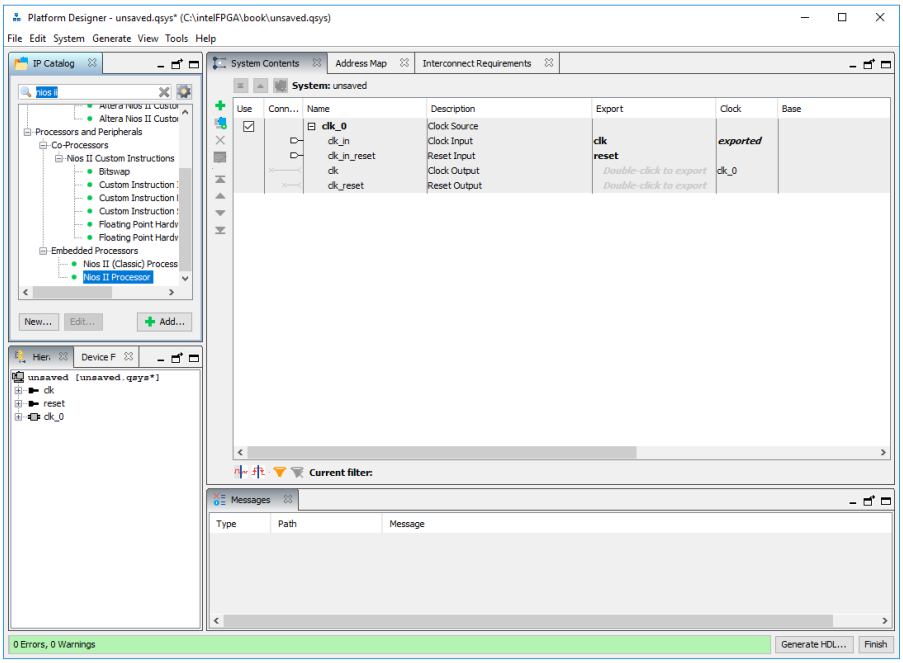

Рисунок 3.7 — Вікно інструменту Platform Designer

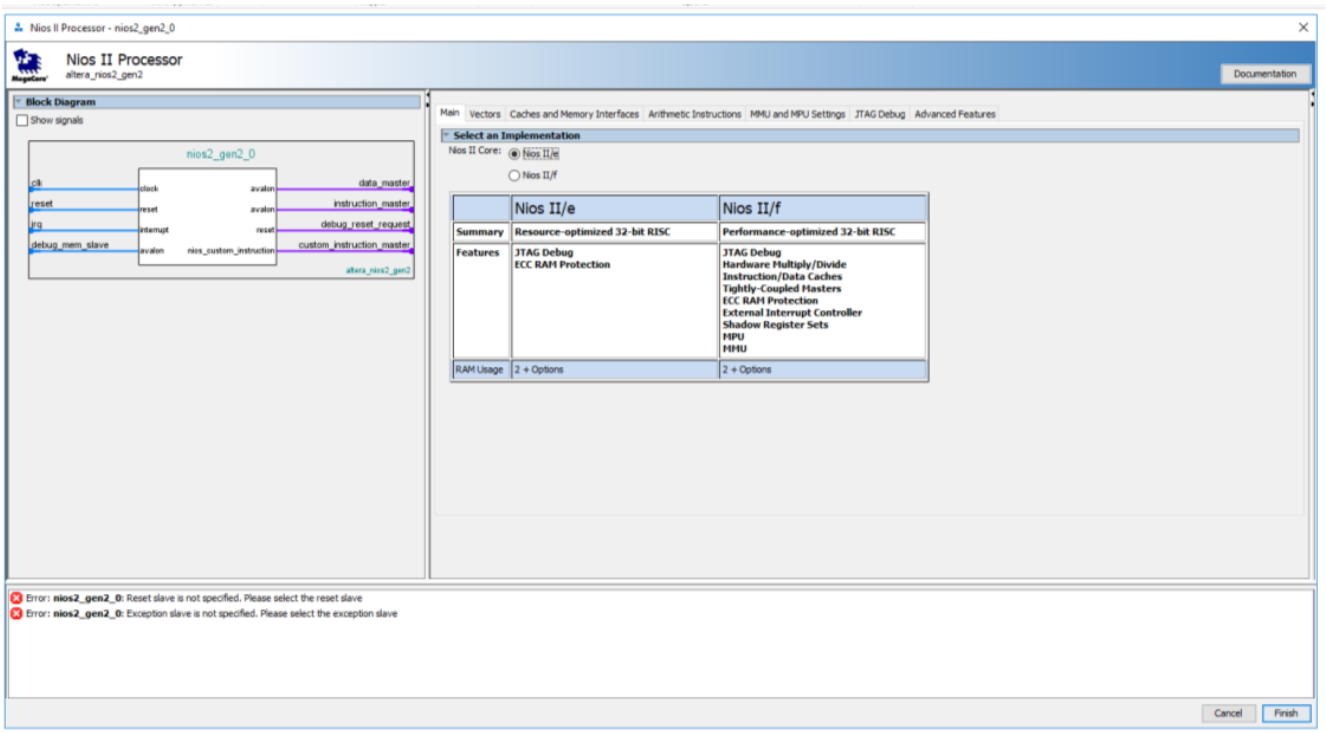

Рисунок 3.8 — Початкове вікно настройки процесора Nios II

Для поточного проекту необхідна вибрати на вкладці Main конфігурацію ядра Nios II / f . На вкладці Vectors визначаються вектора скидання і винятків, проте ми ще не додали пам'ять в систему, тому заповнимо це поле пізніше. На вкладці Cache and Memory Interfaces налаштовуються параметри кешу, на вкладці

Arithmetic Instructions розташовані настройки способів реалізації арифметичних операцій в процесорі; функції на цих двох вкладках доступні тільки при використанні конфігурації Nios II / f. Аналогічна ситуація з настройками блоків управління і захисту пам'яті на вкладці MMU and MPU Settings. На вкладці JTAG Debug за замовчуванням доданий модуль налагодження через інтерфейс JTAG. Більш просунуті налаштування доступні на вкладці Advanced Featured і не розглядаються. Натискаємо кнопку Finish, щоб додати блок в систему. Натисніть правою кнопкою мишки на ім'я блоку і перейменуйте його в cpu.

Додавання модуля пам'яті :

До процесору Nios II можна підключити різні модулі пам'яті, в даному проекті ми скористаємося вбудованою пам'яттю (on-chip memory). В каталозі IP-ядер в категорії Basic Functions> On Chip Memory виберемо On-Chip Memory (RAM or ROM). Наше додаток буде займати небагато місця в пам'яті, у вікні налаштувань блоку (рис 3.9) вкажемо розмір з запасом - 20480 Байт. Інші налаштування збережемо за замовчуванням. Натисніть правою кнопкою мишки на ім'я блоку і перейменуйте його в onchip_mem.

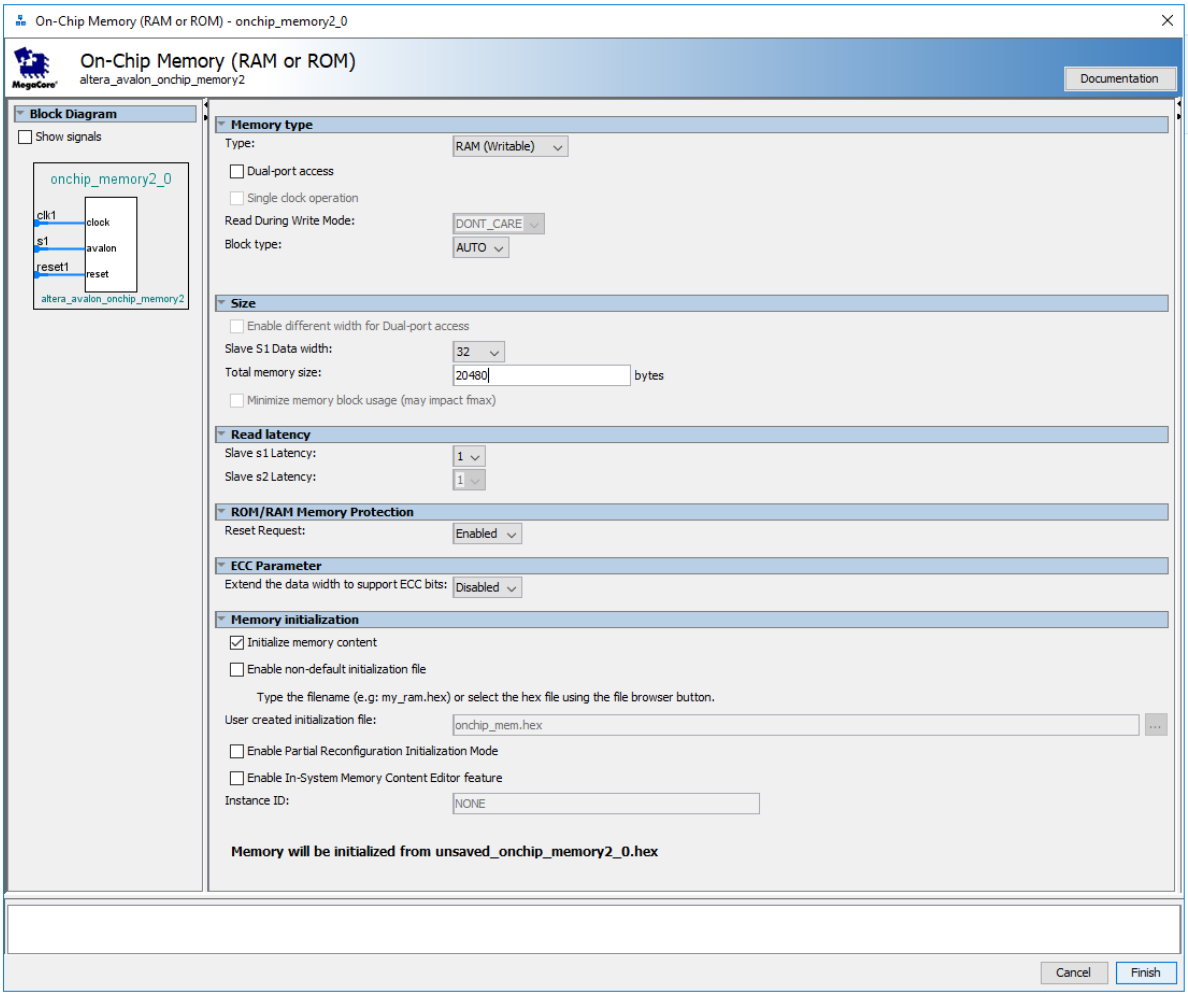

Рисунок 3.9 — Вікно налаштувань блоку On-Chip Memory (RAM or ROM)

Додавання модулів периферії :

У поточній системі в якості периферії використовуються світлодіоди, тобто засоби виведення інформації, і перемикачі, тобто засоби введення інформації. Для підключення даних елементів необхідно скористатися IP-ядром PIO (Parallel Input / Output), яке можна знайти в категорії Processors and Peripherals> Peripherals. В налаштуваннях в обох випадках необхідно вказати бітность підключається шини і напрямок руху даних.

Під час налаштування IP-ядра для перемикачів (рис 3.10 зліва) необхідно вибрати напрямок (Direction) Input. Інші настройки в поточній збірці залишаться за замовчуванням і будуть розглянуті пізніше. Натисніть правою кнопкою мишки на ім'я блоку і перейменуйте його в switch.

В случае настройки для светодиодов (рис 3.10 справа) выбирается направление Output. Остальные настройки остаются по умолчанию. В поле Output Port Reset Value определяется значение, которое будет выставлено на эту шину при инициализации. Нажмите правой кнопкой мышки по имени блока и переименуйте его в ledr.

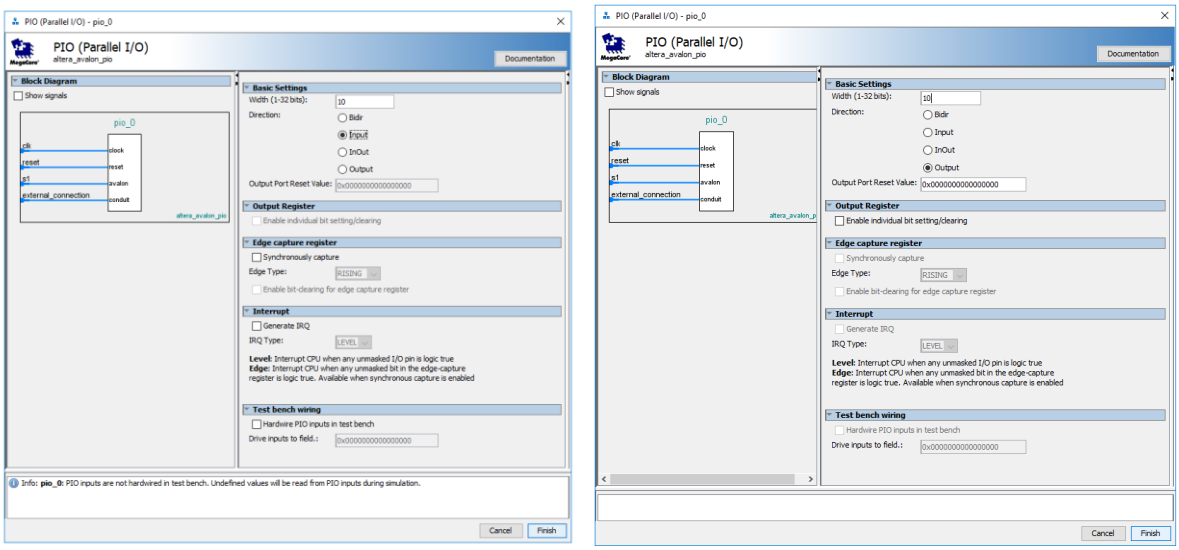

Рисунок 3.10 — Параметри блоків PIO для перемикачів (зліва) і світлодіодів (праворуч)

Визначення міжз’єднань і експорт портів :

Після додавання і налаштування всіх необхідних для даного проекту модулів ми можемо помітити, що у вкладці System Contents блоки пов'язані між собою, а в нижній вкладці Messages знаходиться велика кількість повідомлень про помилку. Для формування зв'язків між блоками необхідно натиснути на порожню точку в місці перетину потенційних ліній зв'язку між блоками.Такі точки можуть бути тільки між шинами одного типу, тобто, редактор не дозволить з'єднати між собою, наприклад, порти тактового сигналу і скидання.

Кожному блоку необхідний тактовий сигнал, тому для початку з'єднаємо джерело тактового сигналу - порт clk блоку clk_0 - з портами для тактового сигналу у всіх блоків.

Аналогічно з сигналом скидання - порт clk_reset блоку clk_0.

Периферія взаємодіє з процесором за допомогою інтерфейсу Avalon MM. Порти введення / виводу не сприймають інструкції і приймають і / або повертають тільки дані, тому їх необхідно з'єднати з портом процесора data_master. У той же час блок пам'яті взаємодіє з процесором через інструкції і передає і приймає дані, тому необхідно з'єднати пам'ять з портом процесора instruction_master.

Щодо проекту збирається система є чорним ящиком з набором входів і виходів. Порти для взаємодії із зовнішніми елементами формуються експортуванням. Прикладом такого експорту є порти clk і reset блоку clk_0, експортовані назовні за замовчуванням. У зібраній системі зовні також підключаються світлодіоди і перемикачі, тому у даних блоків необхідно експортувати порти. Для цього навпроти порту external_connection необхідно двічі клацнути мишею в стовпці Export, після чого дати ім'я даного порту. Призначимо імена аналогічно назвою блоків. Результат виконаних операцій представлений на (рис 3.11).

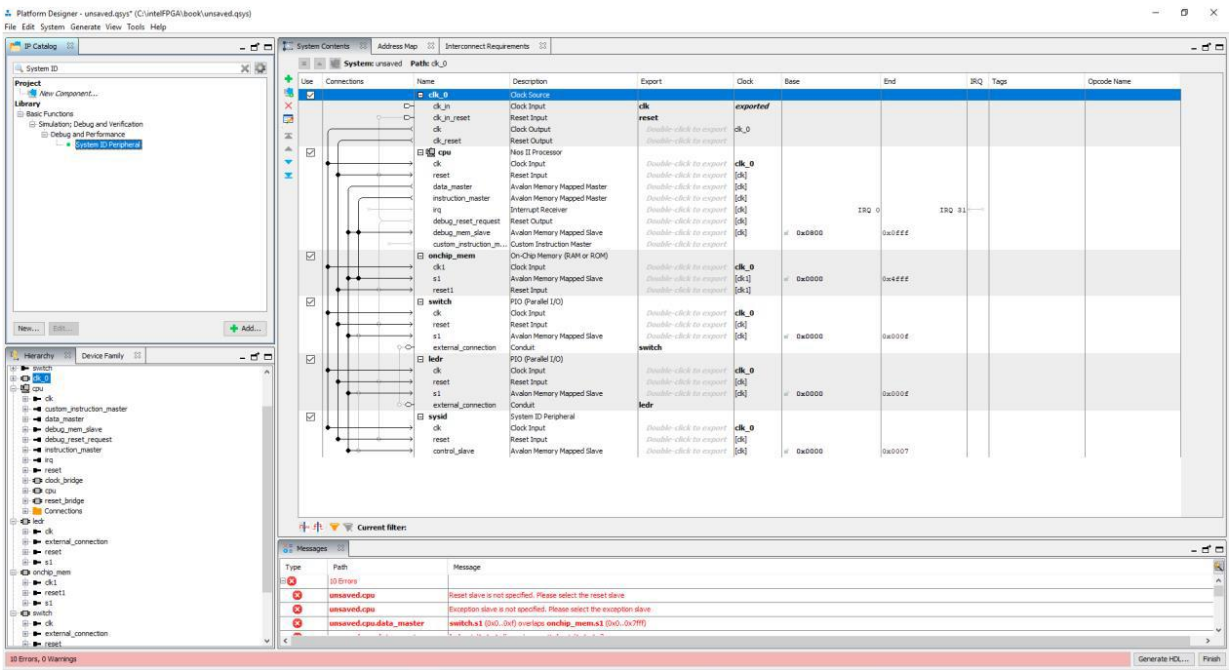

Рисунок 3.11 — Схема міжз’єднань в системі на базі Nios II

Призначення базових адрес і визначення векторів скидання і винятків :

Для доступу до периферії процесор використовує унікальний базовий адресу пристрою. Переглянути та змінити базову адресу кожного пристрою можна в

Platform Designer у вкладці Address Map. За замовчуванням при додаванні кожного пристрою використовується стандартна область адрес, однак перетин адресних просторів між собою не допускається, про що свідчать помилки в нижній частині екрана (рис 3.11). Platform Designer дозволяє як відредагувати базові адреси вручну, так і автоматично розподілити адресний простір між елементами системи. Для цього необхідно в меню System вибрати Assign Base Addresses. Результат автоматичного розподілу представлений на (рис 3.12).

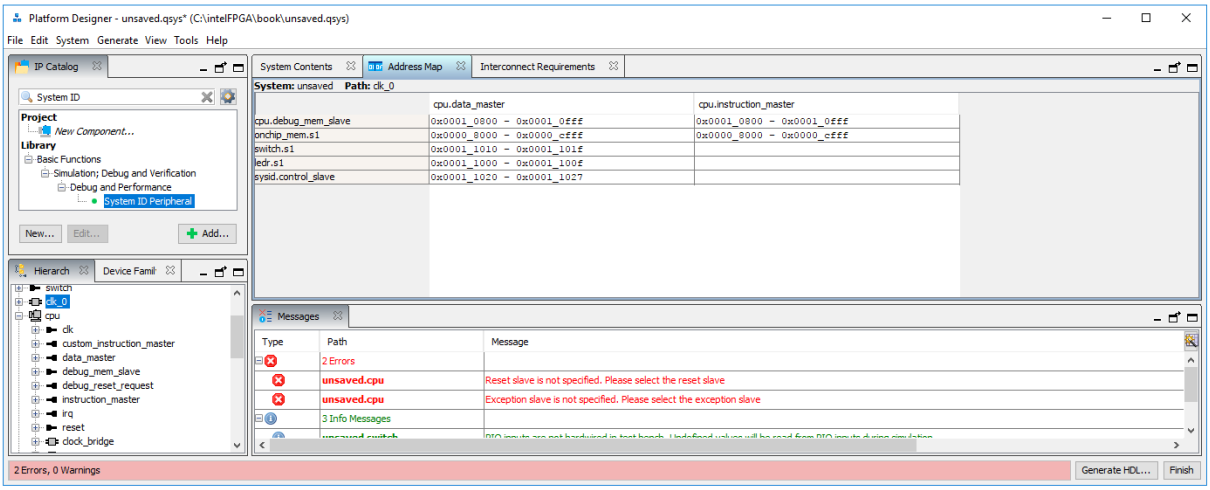

Рисунок 3.12 — Адресний простір системи

Тепер, коли ми додали в нашу систему блок пам'яті і розподілили адресний простір, необхідно призначити вектора скидання і виключення в процесорі. В налаштуваннях процесора Nios II на вкладці Vectors в полях Reset Vector Memory і Exception vector memory в випадаючому списку вибрати onchip_mem.s1.

Далі необхідно зберегти зібрану систему натисканням сполучення клавіш Ctrl + S або через меню File> Save, назвати систему nios_load1.qsys і зберегти її в каталозі проекту.

Процесс создания системы почти завершён. Нам осталось лишь сгенерировать файлы для синтеза системы в среде Quartus, т.е. сформировать HDL-код и bsf-файл. Для этого нажмем кнопку Generate HDL…. В появившемся окне необходимо выбрать язык синтеза, в нашем случае это VHDL, поставить галочку Create block symbol file (.bsf) и нажать клавишу Generate (рис 3.13). Після

завершення генерації і появи повідомлення про успішну генерації всіх файлів можна закрити інструмент Platform Designer.

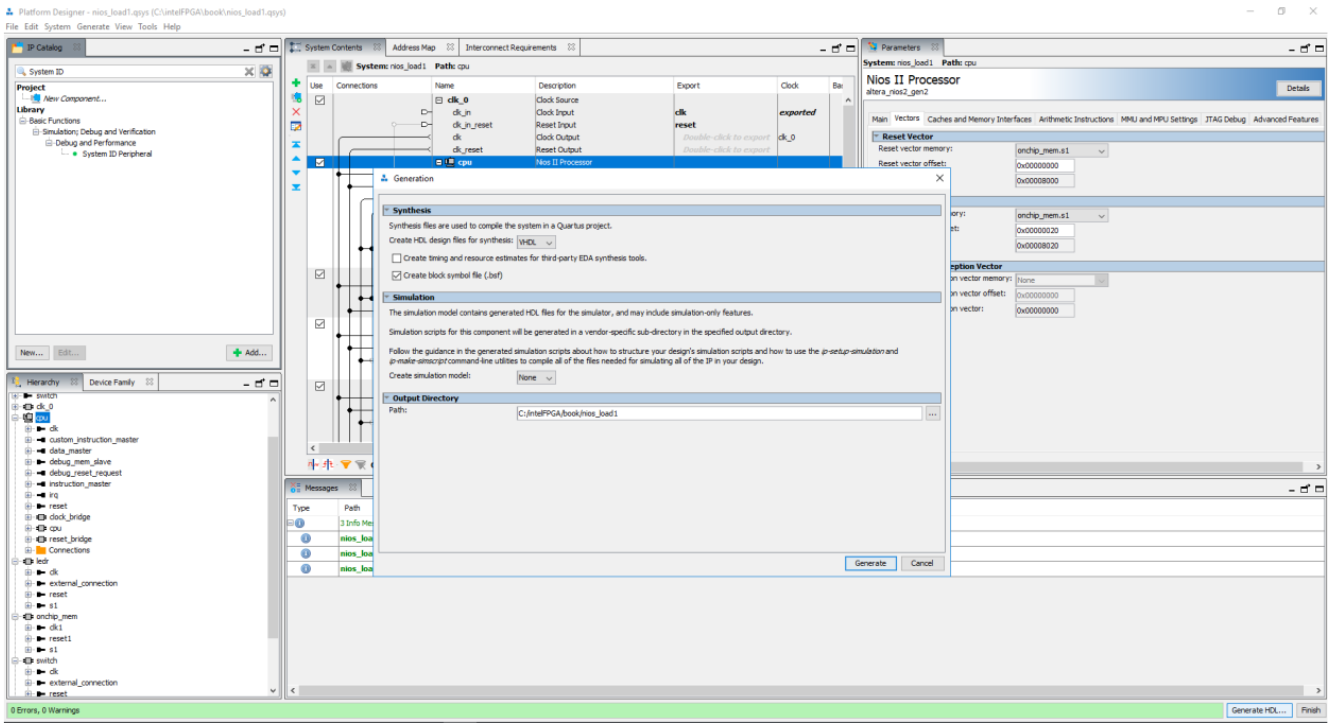

Рисунок 3.13 — Вікно настройки параметрів генерації вихідних файлів HDL в Platform Designer

Platform Designer генерирует следующие файлы :

• nios_load1.qsys: файл дизайну, що містить конфігурацію системи. Його можна розглядати як «вихідний файл», який можна використовувати за допомогою Platform Designer для відновлення інших файлів.

• nios_load1.sopcinfo: містить необхідну інформацію про конфігурацію, використовується Nios II EDS для генерації BSP.

• nios_load1.vhd: Це файл VHDL верхнього рівня для створеної системи Nios II.

• Інші файли VHDL: це файли VHDL для модулів введення / виведення і підсистем процесора Nios II. Файли onchip_mem.vhd, switch.vhd і ledr.vhd призначені для модулів пам'яті і введення / виводу системи Nios II, їх вміст можна переглянути в текстовому редакторі. Однак код, що описує структуру ядра самого процесора Nios II, обфусцірован.

Після закриття перевірте, що система nios_load1.qsys додана в проект. Якщо в списку файлів вона відсутня, додайте її вручну через меню Project> Add / Remove Files in Project ..., у вікні в категорії Files необхідно натиснути клавішу і у вікні виділити додається файл, після чого натиснути Відкрити. Далі необхідно натиснути кнопку Add і завершити додавання прийняттям змін клавішами Apply і Ok. Після цього можна приступати до створення файлу верхнього рівня.

3.2.3 Реалізація файлу верхнього рівня

HDL-файли, згенерувати за допомогою Platform Designer, можна використовувати і обробляти як звичайні текстові файли. Нам просто потрібно створити систему верхнього рівня з Nios II і включити відповідні файли в проект Quartus. Altera надає можливість реалізувати файл верхнього рівня як у вигляді схеми за допомогою графічного редактора або у вигляді HDL-файлу верхнього рівня.

При використанні першого способу необхідно створити файл формату * .bdf (block diagram file), для цього вибираємо File> New ...> Block Diagram / Schematic File. У вікні програми Quartus відкриється редактор блок-діаграм. Двічі клацнувши по полю редактора, ми відкриємо список доступних графічних блоків для додавання і «конструювання» файлу. За замовчуванням створений графічний блок не додається до складу проекту, тому його необхідно вибрати для додавання вручну, натиснувши клавішу. Файл bsf згенерований в папку з назвою системи на базі Nios II - в каталог nios_load1. Блок зібраної системи відобразиться у відкритому вікні, і ми бачимо, що виходами блоку є тільки ті порти, які ми експортували (рис 3.14).

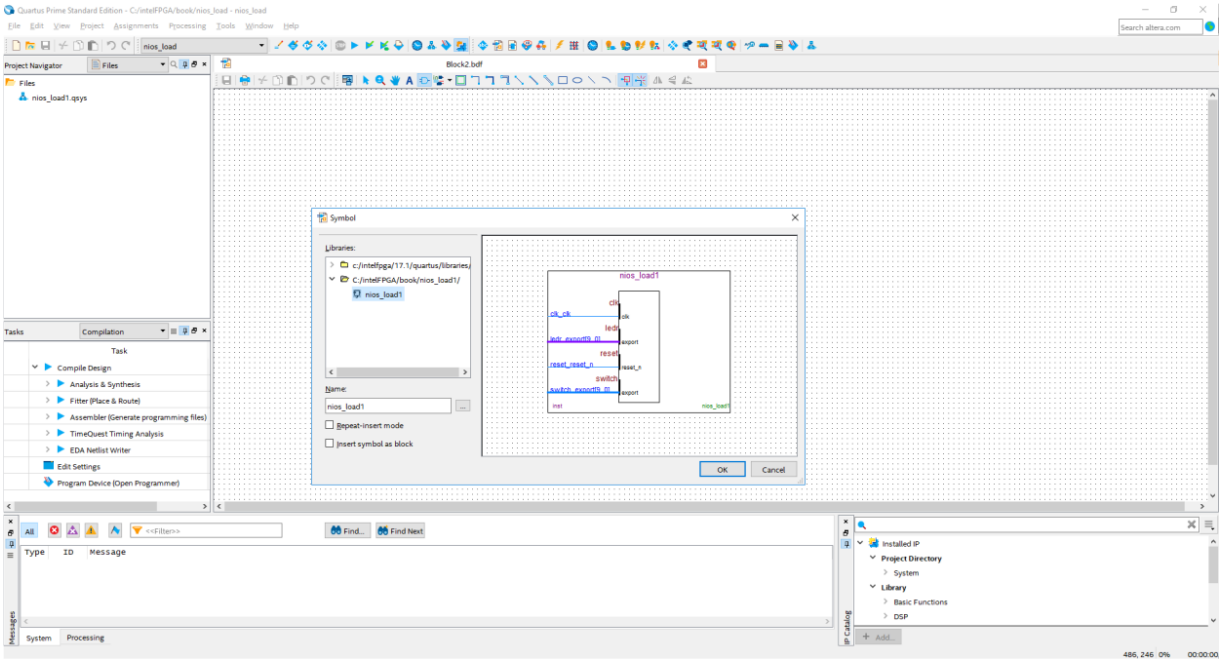

Рисунок 3.14 — Додавання системи на основі Nios II на блок-діаграму

Тепер необхідно додати інформацію про периферію. Для цього додамо на блок-діаграму висновки (на сленгу «Піни» від англ. «Pins») за допомогою інструменту Pin Tool (рис 3.15).

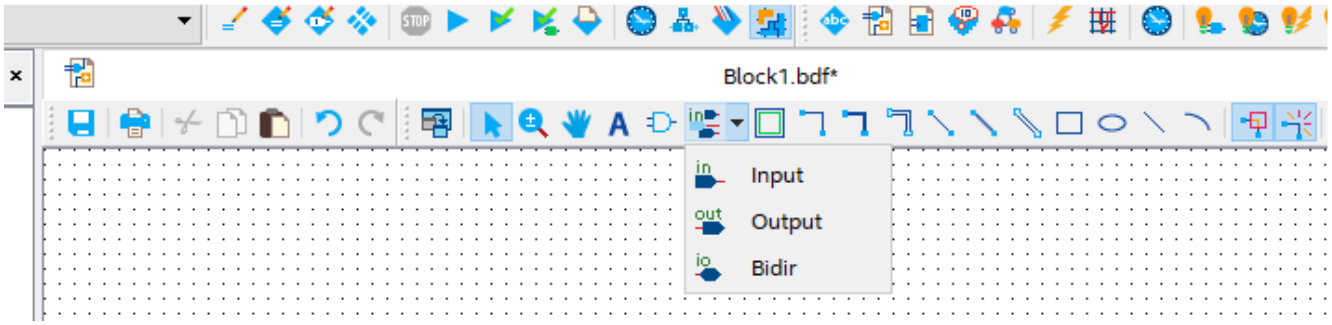

Рисунок 3.15 — Панель інструментів Pin Tool

Далі необхідно зберегти цей файл з назвою nios_load1_top і вказати його в якості верхнеуровневого, для цього в списку файлів натиснути по ньому правою кнопкою миші і вибрати пункт Set as Top-Level Entity.

У другому способі створення файлу верхнього рівня необхідно виконати аналогічну операцію, описавши її мовою VHDL. Оголошення об'єкта верхнього рівня можна знайти в файлі верхнього рівня HDL nios_load1.vhd, а ім'я об'єкта - nios_load1. Цей файл містить декілька проектних одиниць і є досить об'ємним. Ми

можемо відкрити файл і використовувати ключове слово nios_load1 розповсюдження оголошення об'єкта.

В нашем случае код для объявления объекта выглядит следующим образом:

```
entity nios_load1 is
port(
clk_clk : in std_logic := '0';
ledr_export : out std_logic_vector(1 downto 0);
reset_reset_n : in std_logic := '0';
switch_export : in std_logic_vector(9 downto 0) := (others => '0')
);
end entity nios_load1;
```

Він показує, що система, на додаток до сигналів тактового сигналу і скидання, містить 10-розрядний вхідний порт і двухбітний вихідний порт. Імена цих портів визначаються іменами модулів, створених в Platform Designer.

Система Nios II може бути представлена у вигляді звичайного компонента HDL і відповідно легко інтегруватися з іншими блоками в проекті. Оскільки наша демонстраційна система не містить додаткової логіки, нам просто потрібно створити блок верхнього рівня для того, щоб «упакувати», «обернути» (англ. «Wrapper») в нього нашу систему з Nios II. Ще раз загострити увагу: сутність верхнього рівня (Top-Level Entity) - такий блок, чиї висновки (входи і виходи) безпосередньо приєднуються до висновків самої мікросхеми, тобто до елементів введення-виведення, «ніжок» або «пінам» (сленг) ПЛІС. Код HDL показаний в (Додатку Б).

Після визначення файлу верхнього рівня необхідно перевірити коректність конфігурацій і підключень, запустивши етап компіляції під назвою Analysis & Synthesis, двічі клацнувши по ньому в списку Compile Design. При успішному завершенні цього етапу компіляції необхідно встановити зв'язок (програмно) між створеними пінами і фізичними ніжками, до яких підключена відповідна периферія.

3.2.4 Налаштування периферії в Pin Planner

Однією з переваг використання ПЛІС перед, скажімо, микроконтроллерами, є можливість конфігурації, в тому числі в реальному часі, різноманітних параметрів елементів введення-виведення. Для даних завдань в якості інструменту, вбудованого в Quartus, ми скористаємося утилітою Pin Planner. В даному навчальному посібнику ми скористаємося тільки однією з багатьох його функцій - ми поставимо відповідність фізичного контакту на чіпі того чи іншого піну в проекті (визначимо властивість Піна «Location»). Запустити Pin Planner можна натисканням по значку на панелі меню, або з головного меню Assignments> Pin Planner. У вікні в центрі зображений вид зверху на розташування входів і виходів кристала, знизу - таблиця пинов проекту, по краях - інструменти настройки відображення інформації на екрані (рис 3.16).

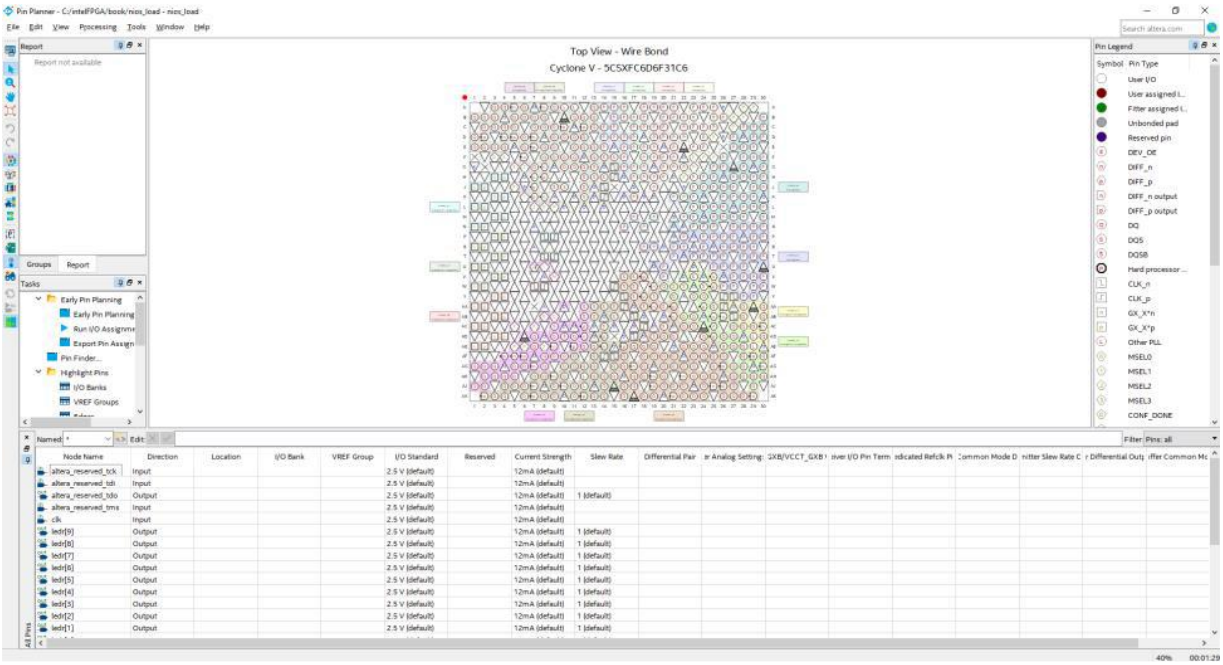

Рисунок 3.16 — Вікно утиліти Pin Planner

У таблиці пінов проекту на початку розташовані чотири з приставкою altera_ - це резервні Піни, генеруються автоматично при створенні Nios II-системи і не вимагають настройки. Іншим пінам необхідно вказати їх розташування в стовпці Location, вписавши туди координати відповідного висновку на кристалі. Ця інформація вказана в документації до плати (таблиця 3.1). Для сигналу скидання

ми скористаємося кнопкою KEY0. Після запису даних в таблицю можна закрити утиліту.

Таблиця 3.1 Відповідність периферії висновків ПЛІС

Назва виведення	Виведення на ПЛІС	Назва виведення	Виведення на ПЛІС
clk	PIN_AF14	reset_n	PIN_AJ4
ledr[0]	PIN_AA24	switch[0]	PIN_AB30
ledr[1]	PIN_AB23	switch[1]	PIN_Y27
ledr[2]	PIN_AC23	switch[2]	PIN_AB28
ledr[3]	PIN_AD24	switch[3]	PIN_AC30
ledr[4]	PIN_AG25	switch[4]	PIN_W25
ledr[5]	PIN_AF25	switch[5]	PIN_V25
ledr[6]	PIN_AE24	switch[6]	PIN_AC28
ledr[7]	PIN_AF24	switch[7]	PIN_AD30
ledr[8]	PIN_AB22	switch[8]	PIN_AC29
ledr[9]	PIN_AC22	switch[9]	PIN_AA30

3.2.5 Компіляція проекту і прошивка плати

Для генерації прошивки для ПЛІС необхідно виконати пункт Assembler зі списку компіляції (Compile Design), для цього двічі клацнемо мишею по цьому пункту. З огляду на те, що аналіз і синтез ми вже запускали і з тих пір дизайн проекту не міняли, компіляція вже почнеться з наступного пункту - Fitter, а потім вже запустить Assembler.

За результатами компіляції згенерує файл прошивки для ПЛІС в форматі * .sof, його назва буде збігатися з назвою файлу верхнього рівня, знайти його можна в підкаталозі проекту / output_files. Для прошивки ПЛІС підключіть плату до харчування і до комп'ютера, запустіть її і запустіть утиліту Programmer натисканням на значок або з меню Tools> Programmer. У вікні розташовані настройки конфігурації ПЛІС (рис 3.17).

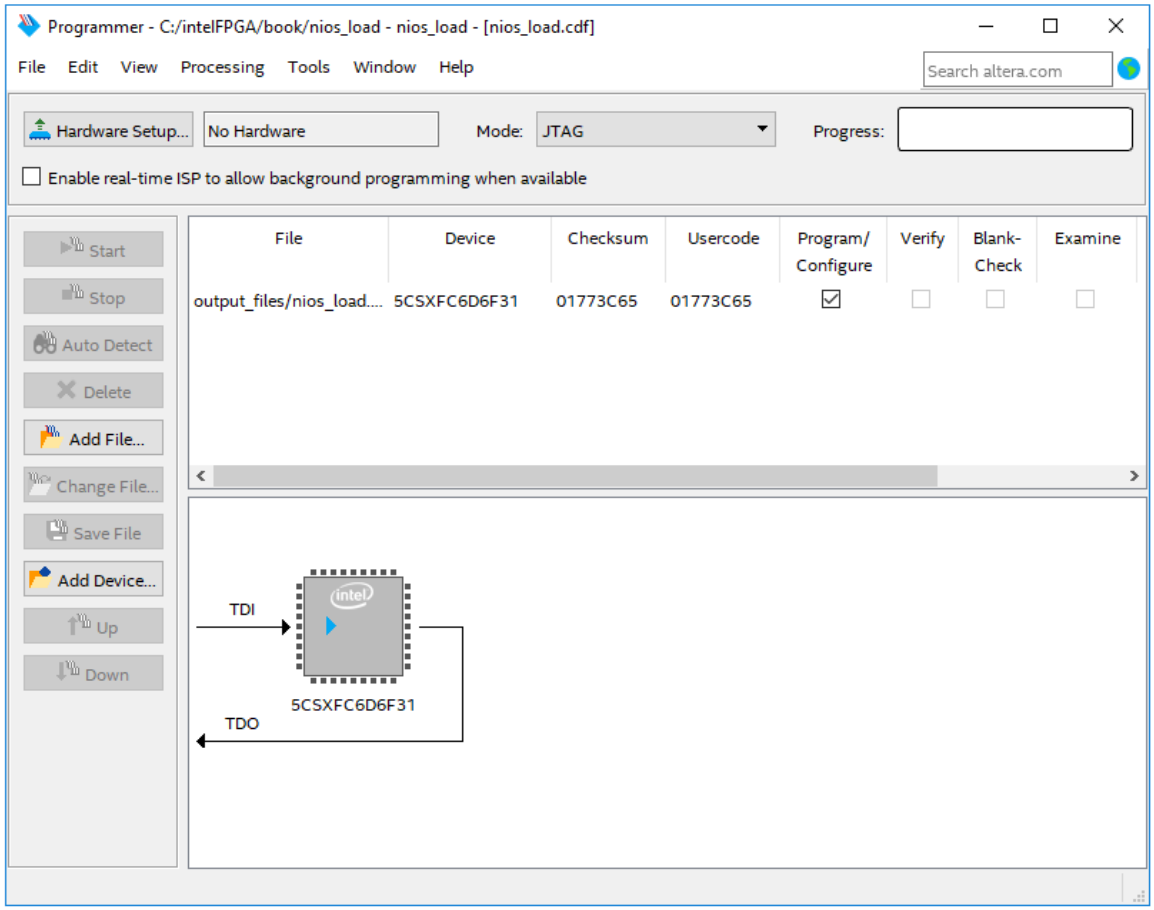

Рисунок 3.17 — Вікно утиліти Programmer

Пристрій, для якого ми розробили дизайн, було вибрано на самому початку при створенні проекту, цю інформацію і сам файл для прошивки утиліта автоматично вибрала з каталогу проекту. Однак частина площі нашої мікросхеми займає апаратна система на кристалі з ядром ARM Cortex-A9, яка називається HPS (hardware processing system). Дана система також може бути налаштована по JTAG інтерфейсу, але наш проект відноситься тільки до частини програмованої логіки мікросхеми, робота з HPS вимагає окремого посібника. Для коректної настройки параметрів прошивки плати необхідно виконати наступні кроки:

1. Натисніть на кнопку Hardware Setup ..., у вікні виберіть підключений до ПК пристрій, натисніть кнопку Close (рис 3.18, а);

2. Видаліть зі списку файл прошивки і натисніть кнопку Auto Detect;

3. У вікні (рис 3.18, б) необхідно вибрати наш пристрій: 5CSXFC6D6, натиснути клавішу OK; в списку підключених пристроїв додасться SOCVHPS - це і є ядро ARM;

4. Виберіть пристрій ПЛІС (5CSXFC6D6), натисніть клавішу Change File ..., у вікні, (рис 3.18, в) виберіть файл прошивки і натисніть Open;

5. У стовпці Program / Configure поставте галочку в рядку пристрої ПЛІС і натисніть кнопку Start для запуску процесу прошивки (конфігурації ПЛІС); процес передачі даних відображається у верхньому правому куті вікна (рис 3.18, г), напис 100% Successful означає успішне завершення процесу.

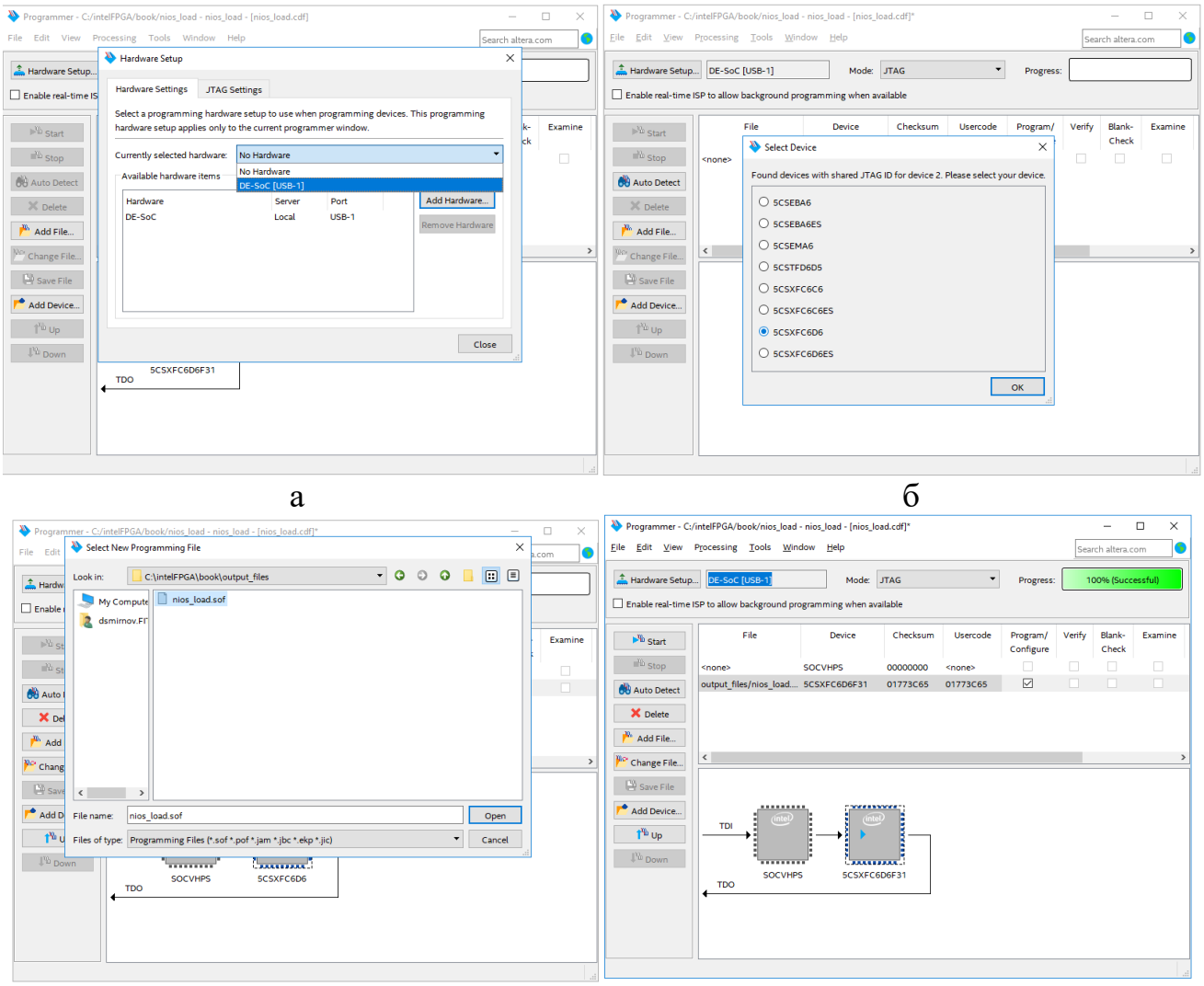

Рисунок 3.18 — Послідовність прошивки ПЛІС за допомогою утиліти Programmer.

В результаті розробки апаратної частини агенту отримані наступні характеристики — ресурсоємності, швидкодії, енергоспоживання

Прогноз ресурсоємності :

+---------------------------------+---+

; Top-level Entity Name ; my_sopc ;

; Family ; Cyclone V ;

; Device ; 5CGXBC7C6F23C7 ;

; Timing Models ; Preliminary ;

; Logic utilization ; < 1 % ;

; Combinational ALUTs ; 719 / 112,960 (< 1 %) // Число задіяних логічних осередків ПЛІС від загального їх числа в кристалі ;

; Memory ALUTs ; 0 / 56,480 (0 %) ;

; Dedicated logic registers ; 550 / 225,920 (< 1 %) ;

; Total registers ; 550 // Число задіяних тригерів ;

; Total block memory bits ; 43,008 / 7,024,640 (< 1 %) // Число задіяних біт пам’яті від загального їх числа в кристалі ;

; Total pins ; 2 / 268 (< 1 %) ;

+---------------------------------+---+

Прогноз швидкодії :

+--+

; Slow 1100mV 85C Model Fmax Summary ;

+-----------+-----------------+--------------------+------+

; Fmax ; Restricted Fmax ; Clock Name ; Note ;

+-----------+-----------------+--------------------+------+

; 235.4 MHz ; 235.4 MHz ; altera_reserved_tck ; ;

+-----------+-----------------+--------------------+------+

+-----------------------------------+---+

Прогноз енергоспоживання :

+----------+

; PowerPlay Power Analyzer Summary ;

+----------+----------+

; Top-level Entity Name ; my_sopc ;

; Family ; Cyclone V ;

; Device ; 5CGXBC7C6F23C7 ;

; Power Models ; Preliminary ;

; Total Thermal Power Dissipation ; 129.44 mW // Загальна потужність споживання ;

; Core Dynamic Thermal Power Dissipation ; 0.35 mW // Дінамічна потужність споживання кристала ПЛІС ;

; Core Static Thermal Power Dissipation ; 100.53 mW // Статична потужність споживання кристала ПЛІС ;

; I/O Thermal Power Dissipation ; 28.55 mW // Потужність споживання блоків введення / виводу кристала ПЛІС ;

+----------+----------+

4. ПРОГРАМНА ЧАСТИНА ДЛЯ РОЗУМНОГО БУДИНКУ

Для розробки програмної частини проекту використовується комплект утиліт, в загальному випадку званий Nios II SBT (Software Build Tools) GUI. Для реалізації необхідно виконати кроки 2, 3 і 4 з пункту 3.3.1. Для розробки програмного забезпечення є два варіанти: розробка за допомогою команд в командному рядку або через графічний інтерфейс. В даному навчальному посібнику використовується другий метод, і в якості графічного інтерфейсу використовується Nios II Eclipse.

4.1 Генерація BSP в Esclipse

1. Запустіть Eclipse: його можна запустити як з меню Пуск, так і з середовища Quartus: Tools> Nios II Software Build Tools for Eclipse; Точно так же Eclipse можна запустити з Platform Designer.

2. При першому запуску з'явиться вікно запиту адреси для робочого простору за замовчуванням;

3. Після ініціалізації середовища виберіть в меню File> New> Nios II Board Support Package;

4. У вікні необхідно дати назву проекту і додати * .sopcinfo файл, який був згенерований нами раніше (рис 4.1);

5. Після обробки файлу системою переконайтеся, що в списку CPU зазначено той процесор, який ви створили в Platform Designer, інші настройки залиште за умовчанням;

6. Натисніть Finish для генерації проекту з бібліотеками BSP.

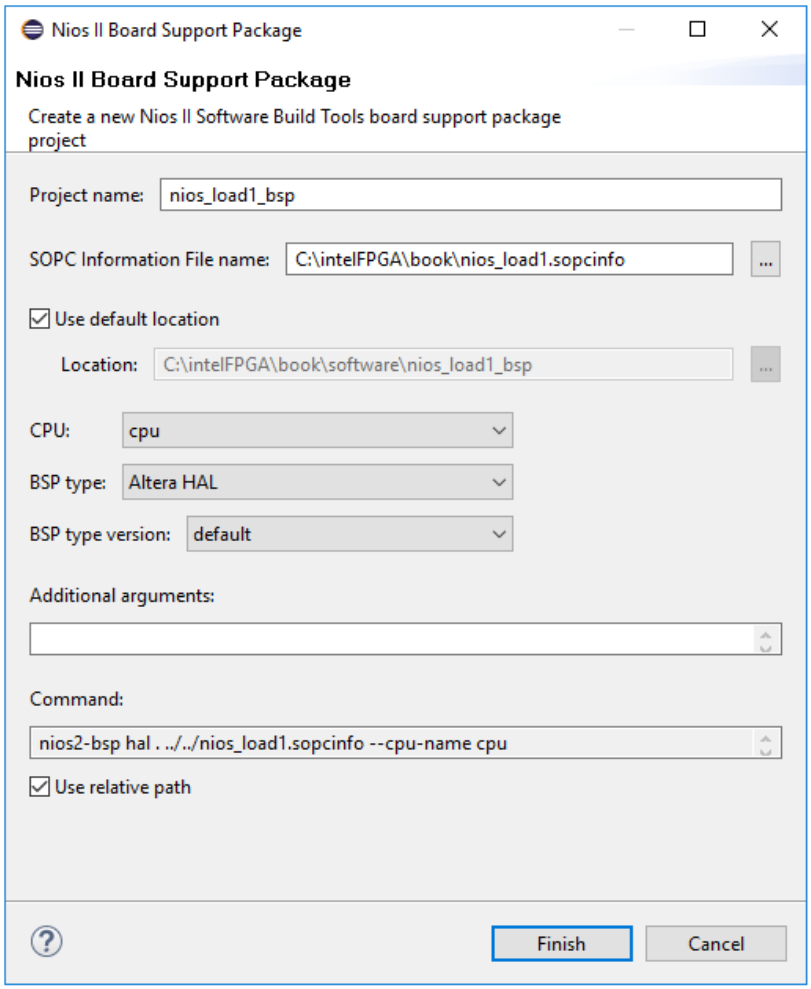

Рисунок 4.1 — Вікно створення BSP

4.1.1 Реалізація програми

Для створення проекту програми необхідно з меню File> New вибрати Nios II Application. У вікні, назвемо проект nios_test1 і вкажемо прив'язку до BSP, створеному раніше (рис 4.2). Решта поля залишимо за замовчуванням і натиснемо кнопку Finish.

Підготовка до створення завершена, тепер необхідно написати код нашого застосування. Для цього необхідно створити файл program.cs в проекті nios_test1, натиснувши правою кнопкою миші на каталог проекту призначеного для користувача програми та вибравши New> Source File. У вікні необхідно задати ім'я файлу з розширенням * .c і вибрати шаблон (Template) <None> для створення абсолютно порожнього файлу. Натисніть Finish. (рис 4.3).

Готовий код вказано в (Додатку Г).

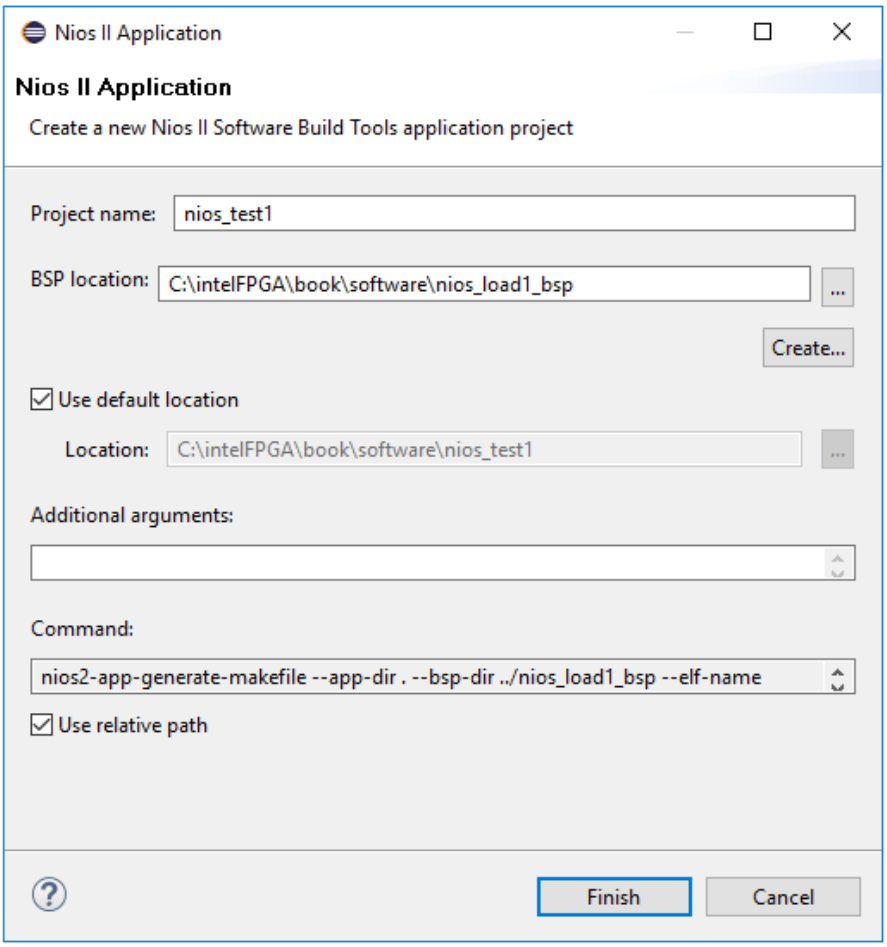

Рисунок 4.2 — Вікно створення проекту програми nios_test1

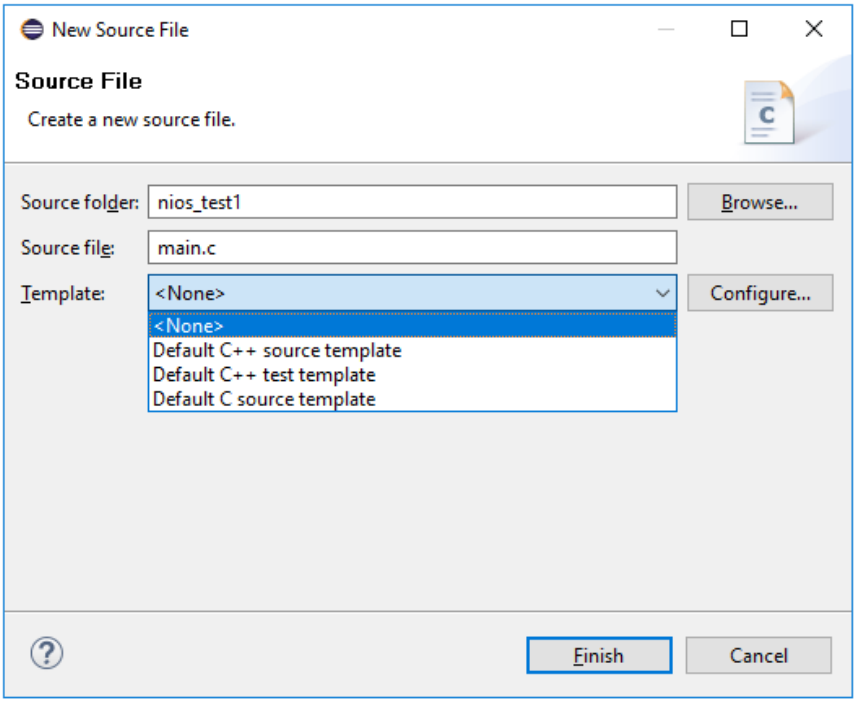

Рисунок 4.3 — Вікно створення файлу програми Program.cs

4.1.2 Компиляція і запуск програми на платі

Після збереження натисніть правою кнопкою миші по каталогу проекту і виберіть Build Project. Прогрес збірки відображається в консолі, при вдалому завершенні компіляції в останніх повідомленнях вказується розмір програми і розмір залишилася вільної пам'яті, а в останньому рядку йдеться про успішне завершення збирання (рис 4.4).Результатом компіляції є набір файлів, ключовий з них для нас на поточний момент - це файл з розширенням * .exe - це файл образу програми.

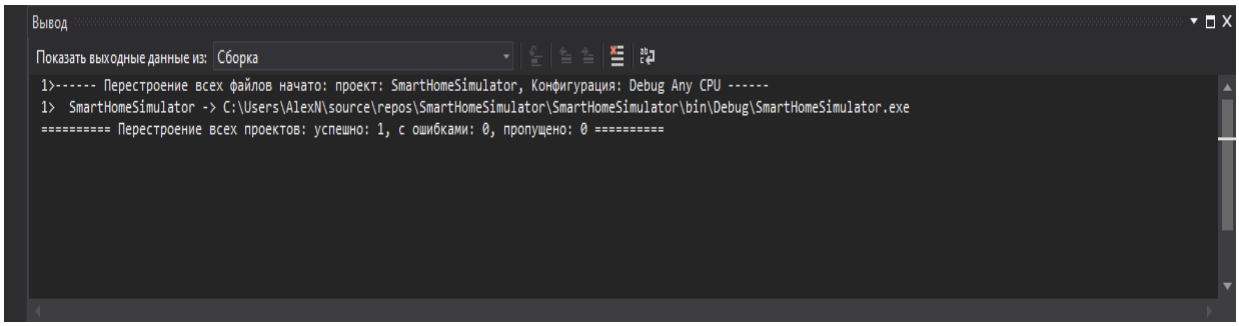

Рисунок 4.4 — Результат збірки проекту

Файл образу необхідно завантажити в ПЛІС, безпосередньо в пам'ять процесора. Для цього необхідно натиснути правою кнопкою мишки по каталогу програми та вибрати Run as> Nios Hardware. При першому зверненні до даної команді відкриється вікно з налаштуванням конфігурації запуску додатка. У вікні на вкладці Project відображається інформація про проект, з яким йде робота і вказується ім'я файлу образу (рис 4.5).

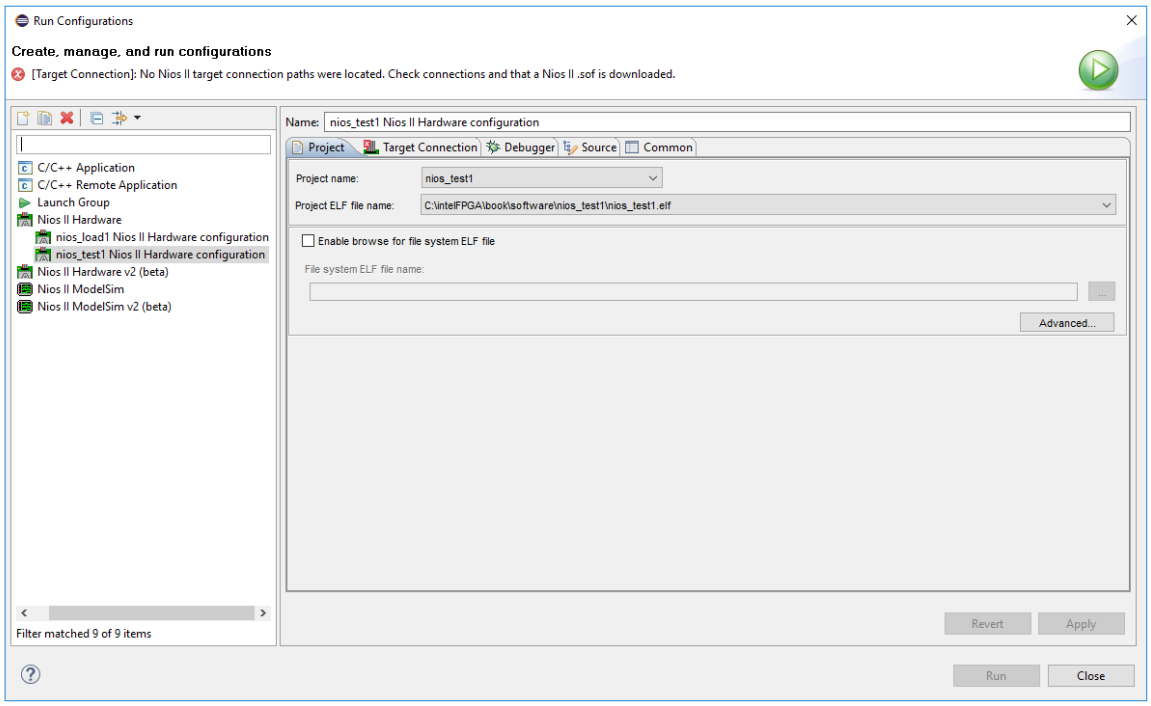

Рисунок 4.5 — Вікно налаштування запуску програми

На вкладці Target Connection відображаються доступні підключення платформи, на яких можна запустити скомпільований додаток. Натиснемо Refresh Connections, щоб підключена плата з системою на Nios II з'явилися

в списку. При появі пристрої в списку можна перевірити збіг System ID і тимчасових міток з тим, щоб упевнитися, що створене додаток зроблено саме для підключеної платформи (рис 4.6).

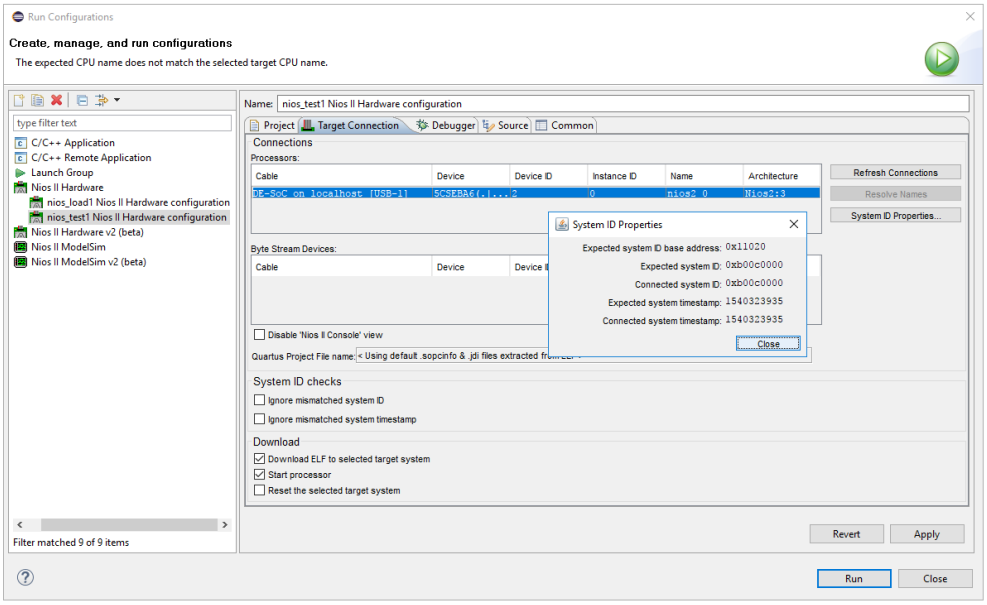

Рисунок 4.6 — Вибір пристрою, що підключається і перевірка його System ID

Залишимо інші настройки за замовчуванням, далі необхідно натиснути клавішу Apply і запустити додаток натисненням кнопки Run. У командному рядку відобразиться процес завантаження даних на ПЛІС, після чого додаток запуститься на платі (рис 4.8).

4.2 Огляд реалізації програми для системи на основі Nios II

Ця програма містить вихідний код SDK для SmartHomeSimulator, який побудований як простий додаток Client-Server. Архітектура системи представлена на наступній схемі (рис 4.7). Цей SDK з'єднується з сервером (симулятором) та інтерфейсує різні функції. SDK реалізований як спільний об'єкт (SO) в Linux і як динамічно пов'язана бібліотека (DLL) у Windows.

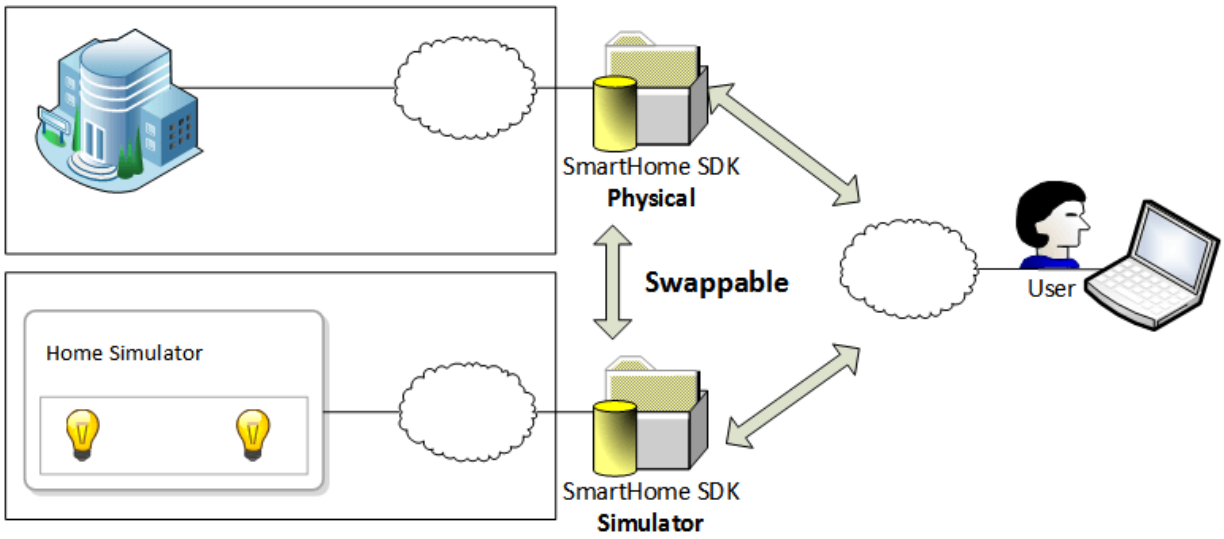

Рисунок 4.7 — Схема роботи програми

Приклад автоматизації

Приклад автоматизації розумного дому пропонує достатньо розмірну модель домену, що містить відносини типу опорного та спадкового для обговорення класової кімнати курсового проекту. Наприклад, розглянемо наступну ієрархію об'єктів у системах.

Розумний дім має кілька поверхів —

На кожному поверсі є кілька номерів, у кожній кімнаті є кілька пристроїв, кожен пристрій може мати кілька функцій, кожен пристрій може бути сполучений з декількома пристроями.

Прикладом функції може бути такий :

- Безпечність
  - Аутентифікація користувача та шифрування
  - Журнали активності
- Безпека
  - Пристрої з підтримкою безпеки із безпечним / небезпечним станом
- Функціональність
  - Увімкнення / вимкнення пристроїв та оновлення статусу
  - Повідомлення про стан, наприклад, поточна температура = 20 ° C
  - Автономні пристрої, наприклад, для ручного керування або несправності
  - Об’єднання пристроїв для включення / вимикання разом

Макет симулятора

Макет симулятора Smart Home можна змінити, відредагувавши файл layout.xml. Цей файл компонування описує домашню конфігурацію, тобто, скільки поверхів знаходиться в будинку, скільки кімнат на кожному поверсі та які пристрої розташовані на якому поверсі. Він також містить інформацію про типи підтримуваних пристроїв та їх парні зв’язки з іншими пристроями.

Підтримувані пристрої

Він підтримує різні пристрої: лампочки, кондиціонер, електросклопідйомники та CD-програвач. Деякі можливості є спільними між усіма пристроями, тоді як інші є залежними від пристрою. Наприклад, лампочку можна просто включити або вимкнути, тоді як CD-програвач можна керувати за допомогою команд наступного чи попереднього треку.

Лампочка

Лампочка – найпростіший пристрій, який можна просто включати або вимикати.

Вікно

Електричне вікно - це пристрій з підтримкою безпеки. У режимі он-лайн та в безпечному режимі його можна відкрити чи закрити, надіславши прості команди. Прийняті команди відкриваються і закриваються, відображаються на цілі значення 0 і 1. Вікно також може повернути статус тексту, що представляє його поточний стан, наприклад, відкритий або закритий.

Кондиціонер

Кондиціонер приймає команди підвищення або зниження потрібної температури. Увімкнувши, кондиціонер намагається досягти бажаної температури шляхом охолодження або нагрівання. Статус і поточну температуру приміщення можна отримати з пристрою.

Плеєр для компакт-дисків

Програвач компакт-дисків - це простий медіапрогравач з кількома вбудованими треками. Увімкнувши живлення, він може вперед, назад шукати трек

або переходити до наступної чи попередньої доріжки. Поточний трек також можна призупинити або відновити. Текстовий стан програвача CD повертає поточну доріжку, що відтворюється.

Будівля

Якщо ви будуєте за допомогою Windows, вам знадобиться MSBuild або Visual Studio, щоб створити файл рішення SmartHomeSimulator.sln. У Linux це рішення можна будувати, використовуючи xbuild та mono утиліти з пакету mono-devel.

```
sudo apt-get install mono-devel

xbuild SmartHomeSimulator.sln

mono bin/Debug/SmartHomeSimulator.exe
```

SmartHomeSDK

Архітектура системи - це проста модель клієнт-сервер. Сервер (симулятор) приймає вхідне з'єднання, зроблене користувачем зі своєї машини. Функціонал підключення керує SDK Smart Home. Цей SDK містить функції взаємодії з тренажером, видаючи різні типи команд та отримуючи відповіді. SDK реалізований як спільний об'єкт (SO) в Linux і як динамічно пов'язана бібліотека (DLL) у Windows. Це дозволяє обміняти SDK варіантом, який спілкується з фактично фізичним розумним будинком замість симулятора :

```
#include <iostream>
#include "SmartHomeSDK.h"

using namespace std;
```

```
int main()
{
  //Smart Home API необхідно ініціалізувати перед будь-якими викликами
  SHAPI_Initialize();

  bool result = SHAPI_SetDevicePoweredOn(1, true);

  //просьба користувача натиснути Enter перед виходом
  cout << "\nPress ENTER to continue...";
  cin.get();

  //Smart Home API повинен бути зупинен в кінці виконання
  SHAPI_Dispose();
}
```

4.3 Опис роботи програми SmartHomeSimulator

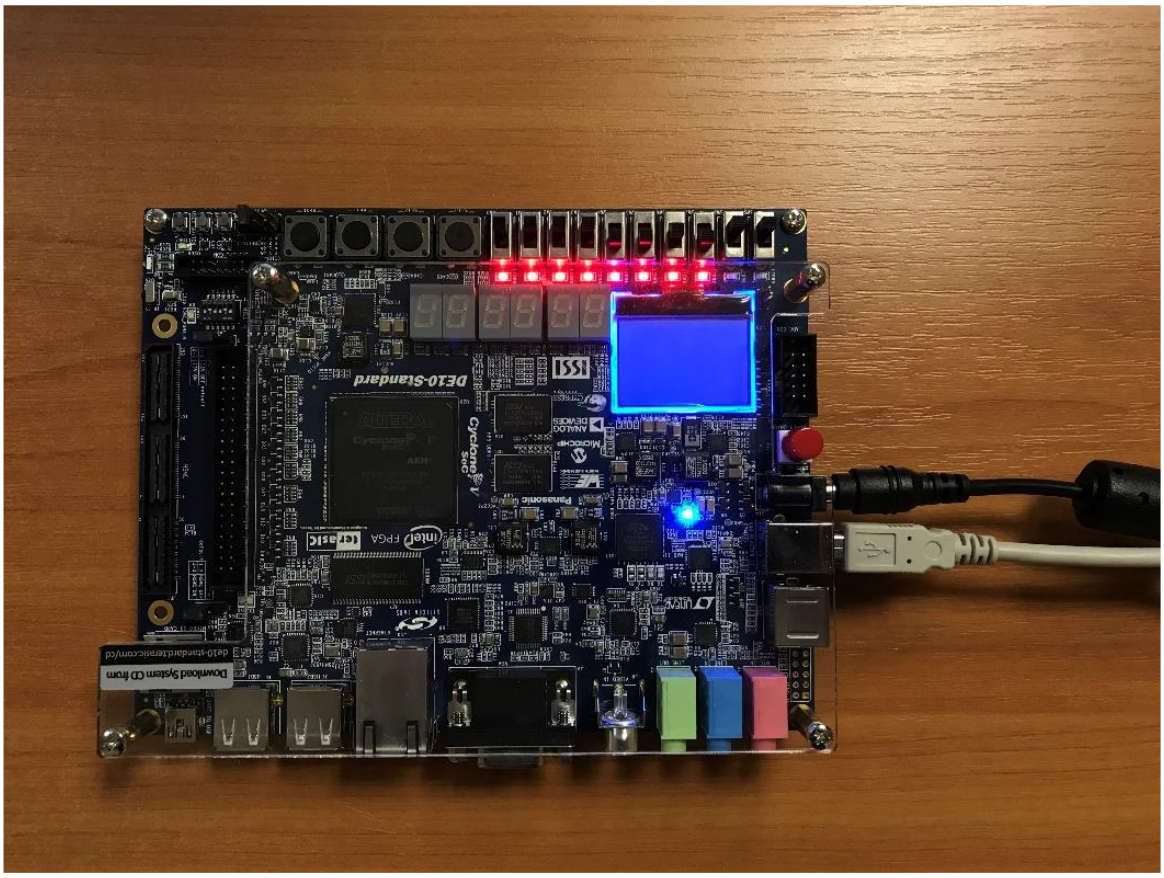

Рисунок 4.8 — Плата з запущеною програмою, фрагмент роботи програми

SmartHomeSDK можна компілювати як в Linux, так і в Windows як 32-розрядні та 64-бітні бібліотеки. Надані Makefiles тестуються на Ubuntu, а також на Windows 10. Виконайте наступні команди.

Спочатку давайте переконаємося, що у нас встановлені необхідні інструменти розробки :

sudo apt-get install make gcc g++

Далі ми складемо спільний об'єкт (SO) і пакуємо його заголовками для випуску. Перша команда нижче очищає будь-які старі спроби збірки. Друга команда виконує компіляцію бібліотеки SDK як спільного об'єкта. Третя команда копіює необхідні файли заголовків і створює пакет випусків у папці випуску :

make clean

make

make headers

Тепер ми можемо протестувати за допомогою наданої програми тестування (рис 4.9). Спочатку переконайтеся, що симулятор SmartHome працює. Потім виконайте такі команди :

cd test

make clean

make

./release/SmartHomeSDKTest.o

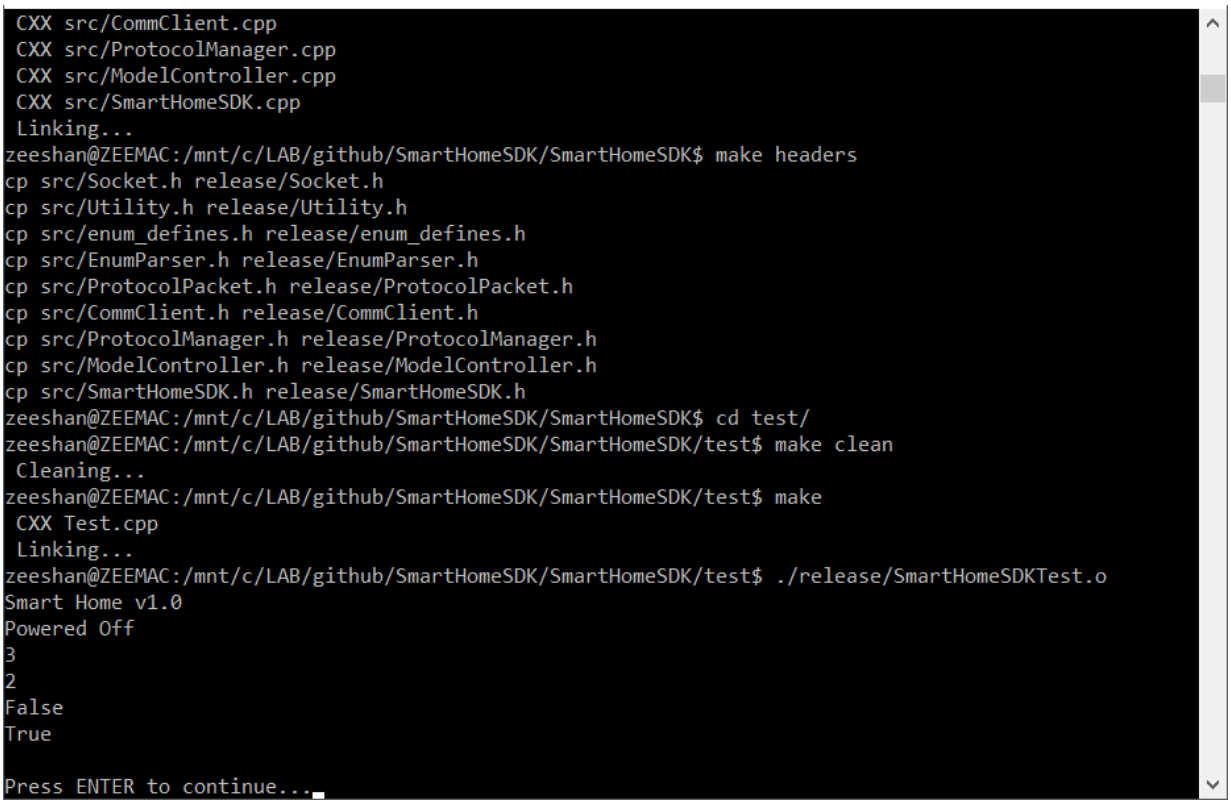

Рисунок 4.9 — Тестування програми

Перша з вищевказаних команд змінює поточний каталог для входу в тестовий підкаталог. Потім виконуємо чисту операцію з видалення тимчасових файлів. Після цього команда make компілює та пов'язує тестову програму у виконуваний файл. Вихід створюється в папці випуску, куди також копіюється файл SO для завантаження. Остання команда виконує тестову програму, яка підключається до тренажера і видає кілька команд для живлення кількох пристроїв, як показано нижче (рис 4.10).

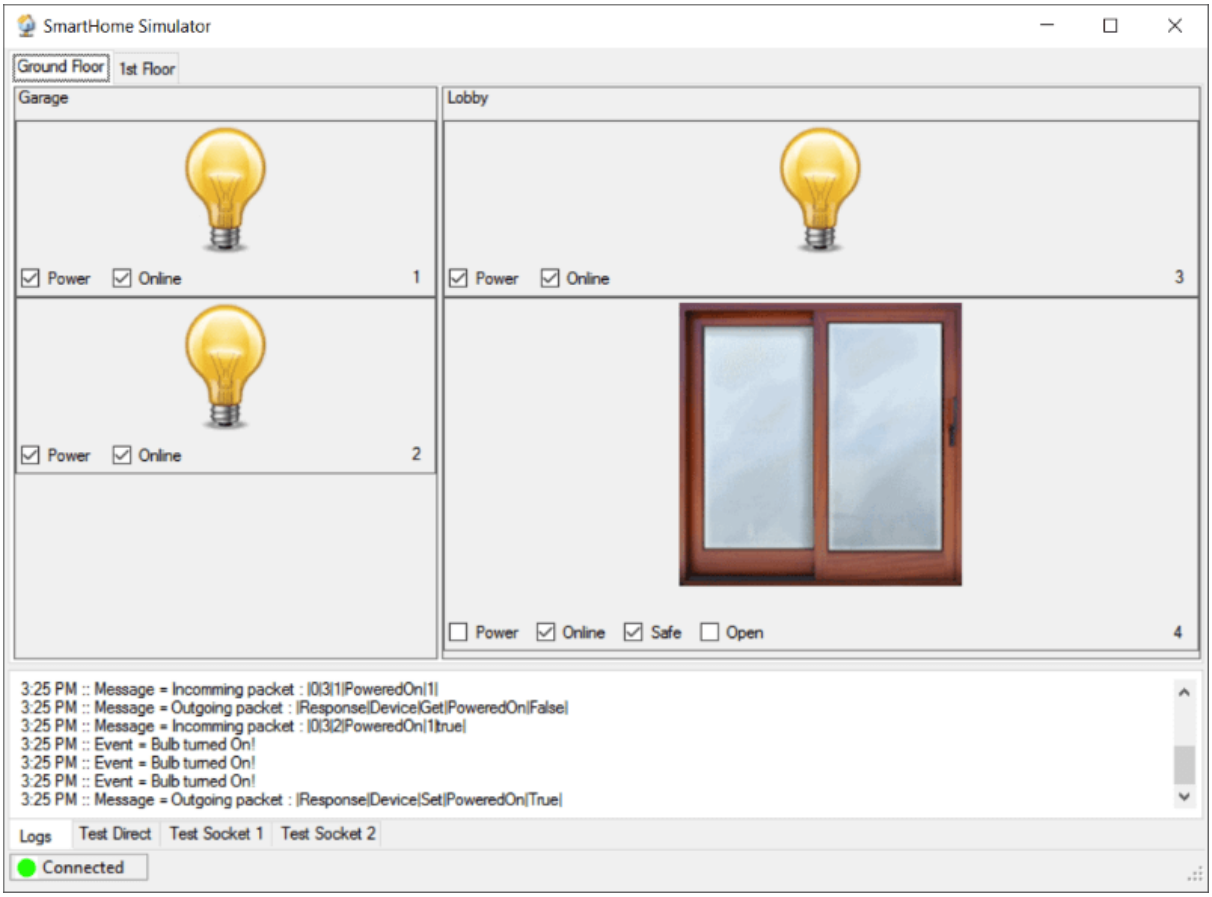

Рисунок 4.10 — Приклад роботи програми

API стилю C

SDK може використовуватися як API стилю C, побудований поверх коду OO. Для цього включайте файл заголовка SmartHomeSDK.h, доступний у випуску. Після включення ви можете викликати викриті функції. Перелік та функціональність цих функцій представлений у ДОДАТКУ B.

4.4 Опис програмного продукту «HomeAssistant»

Система домашньої автоматизації **HomeAssistant** — являє собою безкоштовну і відкриту програмну платформу для комплексного управління домашньою автоматикою, а також для інформаційної підтримки життєдіяльності. Дана система може бути встановлена практично на будь-який комп'ютер (рис 4.11). (на платформі Windows і Linux) і абсолютно не вимоглива до ресурсів. навіть без прив'язки до обладнання вона може бути використана для організації персонального інфо-середовища (рис 4.12).

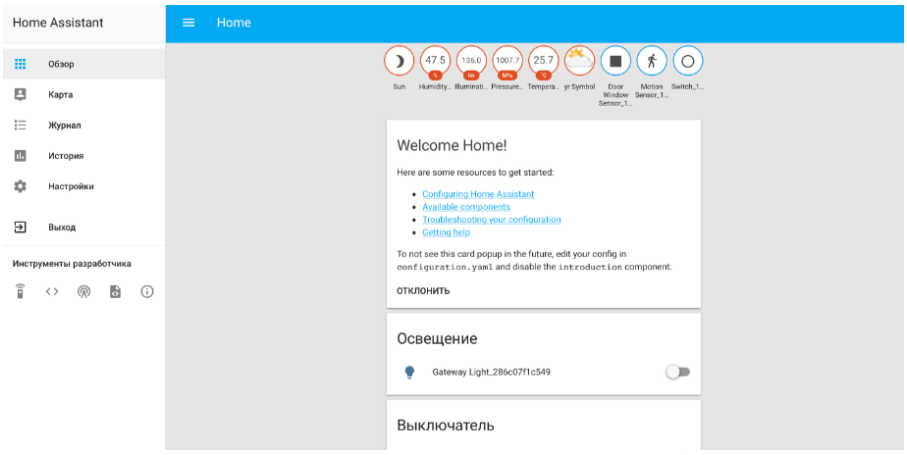

Рисунок 4.11 — Головна сторінка користувача

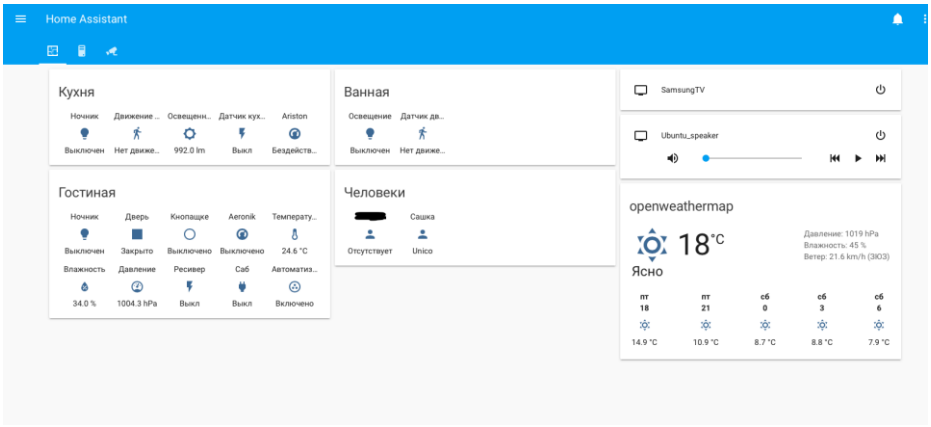

Рисунок 4.12. — Інтерфейс керування умним домом HomeAssistant

Основні можливості:

- проста і швидка установка;
- кросплатформенность (Windows / Linux);
- безкоштовна для особистого або комерційного використання;
- велике і активне співтовариство навколо проекту;
- підтримка різного устаткування;
- багатомовний інтерфейс (Російська / English);
- веб-доступ з будь-якого пристрою;
- веб-інтерфейс з оновленням в реальному часі;
- GPS-трекінг і реакція на місце розташування користувачів;
- голосові повідомлення і розпізнання голосу;
- Push-повідомлення;
- інтеграція зі сторонніми веб-сайтами та сервісами;
- контроль медіа;
- модель безпеки з розмежуванням доступу між користувачами;
- CloudSync - хмарна синхронізація і простий доступ з будь-якого місця;
- система оновлень в один клік;
- побудована на веб-технологіях (PHP / JS / HTML5);
- ООП в реальному житті: класи / об'єкти / властивості / методи;
- програмування за допомогою PHP і / або візуальної середовища Blockly;
- розширений аналіз стану та самодіагностика;
- магазин додатків;
- підтримка динамічних 3d- сцен (WebGL).

HomeAssistant являє собою відкриту платформу для створення систем класу «Розумний будинок». Установчий пакет системи не є готовим продуктом, а, в більшій мірі, є базою, на якій може бути побудований центр управління розумним будинком. Користувачеві системи пропонується самостійно інтегрувати наявні пристрої, а також налаштувати під себе інтерфейс управління і сценарії взаємодії наявних пристроїв.

Платформа пропонує гнучку середу для організації управління і контролю всіх автоматизованих систем, а також засоби створення взаємопов'язаних сценаріїв поведінки компонентів.

З точки зору технічної реалізації, платформа являє собою веб-систему (веб-сайт), що встановлюється на домашній сервер, а також набір додаткових компонентів, що запускаються паралельно з веб-системою.

Для написання внутрішніх сценаріїв системи використовується мова програмування PHP.

Система HomeAssistant об'єднує в собі різні компоненти, дія, багатьох з яких пов'язане з читанням або змінювати будь-які даних. Для організації ефективного обміну даними між різними частинами системи була створена об'єктна модель. Дана модель багато в чому відповідає парадигмі Об'єктно Орієнтованого Програмування (ООП). Вбудована в систему модель досить спрощена і може застосовуватися без глибокого знання будь-якої мови програмування.

У різних частинах системи існує функціонал «прив'язки» того або іншого елемента з об'єктом, його властивістю або його методом. Об'єкти є основою зберігання даних системи, а також описом функцій роботи з цими даними. Саме тому більшість модулів, так чи інакше, посилаються на об'єкти.

Наприклад, елемент меню типу Вимикач використовує пов'язане властивість для зберігання даних про своєму останньому стані, а так само метод об'єкта, як дію, яке треба виконати після зміни стану. З іншого боку, модулі роботи з обладнанням так само використовують пов'язані властивості і об'єкти для зберігання даних, отриманих від відповідних електронних пристроїв.

Наприклад, прив'язавши властивість якогось об'єкта до топіку вимикача в модулі MQTT можна звертатися до цієї властивості для отримання останнього стану фізичного вимикача, а так само використовувати його для установки значення (включення навантаження), таким чином, створюється прозора двосторонній зв'язок між фізичним пристроєм і об'єктом системи HomeAssistant. Одне властивість об'єкта може бути прив'язане до кількох елементам, так, якщо розглядати попередні два приклади, то можна об'єднати їх в один, коли і для

прив’язки вимикача в меню і для прив’язки топіка вимикача MQTT ми використовуємо одну властивість. У такому випадку ми отримуємо керований чер ез меню фізичний вимикач.

У системі передбачена можливість налаштовувати реакцію системи на команди у вигляді тексту або навіть голосові повідомлення (при використанні окремого додатка). В панелі управління для цього призначений модуль «Шаблони поведінки», в якому можна налаштувати не тільки прямі команди, але і досить складні розгалужені діалоги.

В основному, взаємодія з користувачем відбувається через веб інтерфейс. Як «точок входу» можна виділити наступні:

— Основний екран (посилання: http: // АДРЕС_СЕРВЕРА /) - загальний екран системи з меню і «домашніми сторінками»;

— Меню (посилання: http: //АДРЕС_СЕРВЕРА/menu.html) - екран управління, адаптований для мобільних пристроїв;

— Домашні сторінки (посилання: http: //АДРЕС_СЕРВЕРА/pages.html);

— Сцени (посилання: http: //АДРЕС_СЕРВЕРА/popup/scenes.html);

— Панель управління (посилання: http: //АДРЕС_СЕРВЕРА/admin.php) – панель настройки компонентів системи;

— Крім того, за допомогою мобільного додатку додатково з'являються можливості голосового управління.

Оптимальним способом взаємодії з системою є використання стаціонарних терміналів на базі планшетів з встановленим мобільним додатком.

4.4.1 Алгоритми роботи системи «HomeAssistant»

Слід розробити алгоритми управління інженерними системами приміщення. А саме, необхідно здійснювати як автоматичне, так і ручне керування освітленням та опаленням.

Найчастіше буває так, що різні побутові прилади має сенс використовувати в різних режимах для економії витрат на електроенергію. Для того, щоб реалізувати

такий сценарій, введемо поняття «Режим економії ». Цей режим використовується, коли в приміщенні довгий час відсутні люди, тому вимоги до підтримання необхідної температури знижені.

В рамках алгоритму управління опаленням використання режиму економії на увазі зниження підтримуваної температури в приміщенні, що дозволить оптимізувати споживання електроенергії. Наприклад, в період відсутності мешканців у приміщенні або в нічний час.

На (рис 4.13) представлена схема алгоритму управління опаленням. як видно зі схеми алгоритм має на увазі два режими роботи. перший повністю автоматичний, коли системі задана необхідна температура і по її досягненні опалення відключається, якщо ж температура знижується, то система знову підвищує температуру до встановлених значень.

В ручному режимі система автоматично не підтримує заданий рівень температури, а одного разу досягає призначеної температури і відключається. Також слід ввести перевірку на включення кондиціонера, щоб в такому випадку система не нагрівала охолоджуване приміщення.

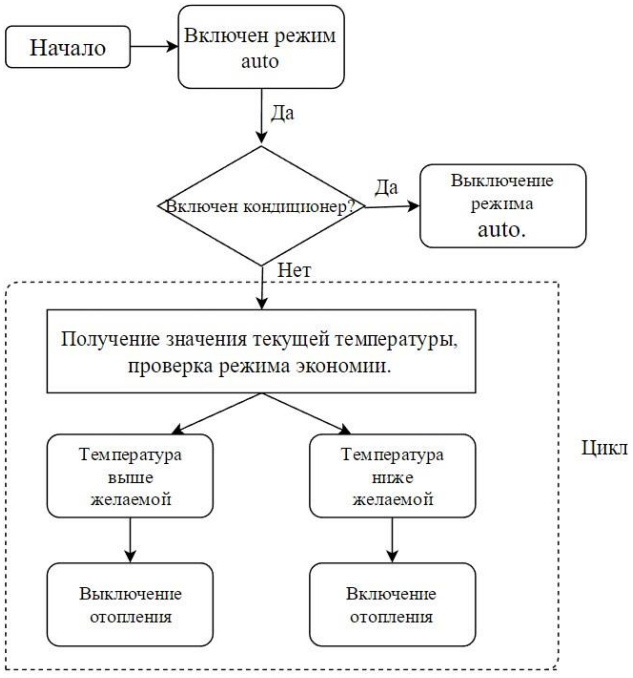

Рисунок 4.13 — Схема алгоритму управління опаленням

Для управління освітленням в автоматичному режимі будуть використані

датчики руху, встановлені в кожному приміщенні.

Так само як і для опалення, в алгоритмі передбачена робота в двох режимах, в автоматичному і ручному.

При роботі освітлення в ручному режимі управління здійснюється шляхом натискання на настінні перемикачі, або на перемикачі на сцені квартири в мобільному терміналі, включення і виключення буде здійснювати завжди, незалежно від освітленості зовні.

В автоматичному режимі система буде самостійно включати освітлення в приміщенні при фіксуванні руху в цьому приміщенні при дотриманні ряду умов. По-перше: необхідно увімкнути автоматичне режиму; по-друге: необхідний недостатній рівень освітленості зовні приміщення. Для реалізації автоматичного вимкнення світла, після включення освітлення буде встановлено таймаут на відключення, якщо в приміщенні буде відсутній рух. Алгоритм роботи освітлення представлений на (рис 4.14).

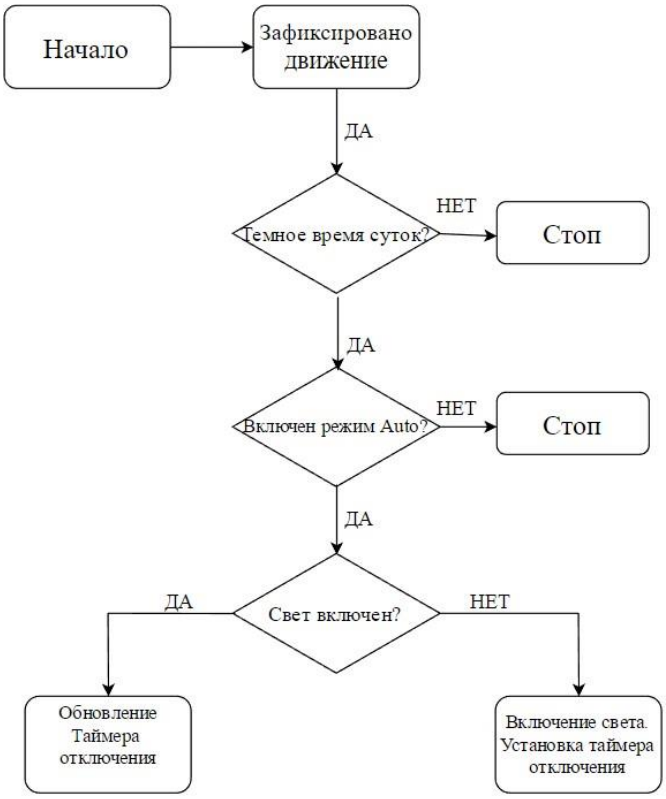

Рисунок 4.14 — Схема алгоритму управління освітленням

4.4.2 Реалізація алгоритмів управління освітленням і опаленням в «HomeAssistant»

Необхідно реалізувати сценарій управління опаленням окремої кімнати, а також отримувати температуру в кімнаті за допомогою встановленого датчика температури.

Для початку потрібно підключити до системи HomeAssistant, датчики температури і вологості, для того, щоб була можливість стежити за їх показаннями в приміщенні. Для цього будуть використовуватися реле sonoff, тому у них є можливість підключення додаткового датчика і відправки його значень по протоколу MQTT.

У самій системі HomeAssistant дані надходять в пункт меню Пристрої -MQTT>, розподілені за відповідними топіками, від представлений на (рис. 4.7).

Home Assistant (далі HA) має вбудований MQTT брокер, але багато хто скаржиться на його нестабільність і обрізання, тому будемо ставити і налаштовувати альтернативний MQTT брокер Mosquitto.

Для початку необходимо обновити стан системи до актуальної. Підключаємося до плати за ssh і виконуємо наступні команди:

sudo apt-get update

sudo apt-get upgrade

Далі ставимо брокер:

sudo apt-get install mosquitto mosquitto-clients

Після установки брокера, необхідно захистити його від підписки кого б то не було за допомогою зв'язки логіна пароля, для цього скористаємося наступною командою:

osquitto_passwd –c /etc/mosquitto/passwd homeassistant

Далі треба буде ввести два рази пароль за запитом. Ця команда створить зв'язку логіна homeassistant і пароля який ви задали в файлі / etc / mosquitto / passwd і тепер нам треба нацькувати брокер на цей файл і заборонити анонімні

підключення до нього. Зробимо це так, відкриємо файл конфіга брокера: sudo nano /etc/mosquitto/conf.d/default.conf

І запишемо туди наступні рядки:

allow_anonymous false password_file /etc/mosquitto/passwd

Зберігаємо файл і перезавантажуємо брокер командою:

sudo systemctl restart mosquitto

Далі можемо перевірити, що все налаштовано правильно. Відкриємо паралельно два вікна терміналу і підключимося в обох до нашої плати по ssh. Далі в одному з них напишемо:

mosquitto_sub –h localhost –t test –u "homeassistant" –P "ваш_пароль"

А в другому:

mosquitto_pu –h localhost –t "test" –m "Test message" –u "homeassistant" -P "ваш_пароль"

Після цього в першому терміналі ми побачимо з'явилося повідомлення Test message.

4.4.3 Налаштування стороннього брокера в HomeAssistant

Відкриваємо основний файл конфігурування HomeAssistant дописуємо в кінець файлу:

```
mqtt:
broker: 192.168.1.100
  port: 1883
  client_id: home-assistant-1
  keepalive: 60
  username: !secret mqtt_login
  password: !secret mqtt_password
  protocol: 3.1
  birth_message:
    topic: "tele/hass1/LWT"
    payload: "Online"
```

```
    qos: 1
    retain: true
  will_message:
    topic: "tele/hass1/LWT"
    payload: "Offline"
    qos: 1
    retain: true
```

Розберемо по порядку:

broker ip адресу брокера в вашому випадку IP адреса малини

client_id унікальне ім'я подключающегося клієнта

keepalive проміжок часу, в через яке клієнт буде слати брокеру повідомлення "я живий"

username ім'я користувача для підключення, задавали вище

password пароль для підключення до брокера, так само задали вище

protocol версія протоколу

birth_message / will_message повідомлення про доступність / недоступність клієнта

topic топік в якому розміщується повідомлення

payload текст повідомлення

qos грубо кажучи пріоритет повідомлення

retain чи повідомлення було надіслано іншим клієнтам

Все що ми набудували - стосується тільки інформації про статус самого HA в брокера

4.4.4 Додаємо пристрій з MQTT

HomeAssistant підтримує досить великий список пристроїв, які можна підключити таким способом. Приклад реле з esp8266 на прошивці Tasmota, яка управляє котлом опалення через сухі контакти (рис. 4.17). Та керує таймером на електронної плиті (рис. 4.15).

Рисунок 4.14 — Приклад налагодження MQTT

Так само давайте розберемо детальніше:

platform ну тут зрозуміло вказуємо що використовуємо MQTT платформу;

name назва світча, яке буде відображатися в HA, воно ж йде в entity;

state_topic топік звідки читаємо стан;

command_topic топік в який будемо передавати команди;

availability_topic топік в якому пишеться доступність пристрою;

qos розглядали вище;

payload_on повідомлення про включення;

payload_off повідомлення про виключення;

payload_available повідомлення про доступність;

payload_not_available повідомлення про недоступність;

state_on статус який буде вважатися включеним;

state_off статус який буде вважатися виключеним;

retain розглядали вище;

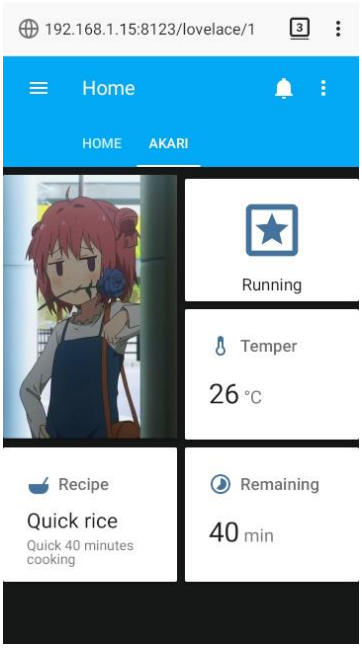

Рисунок 4.15 — Приклад роботи Брокера MQTT

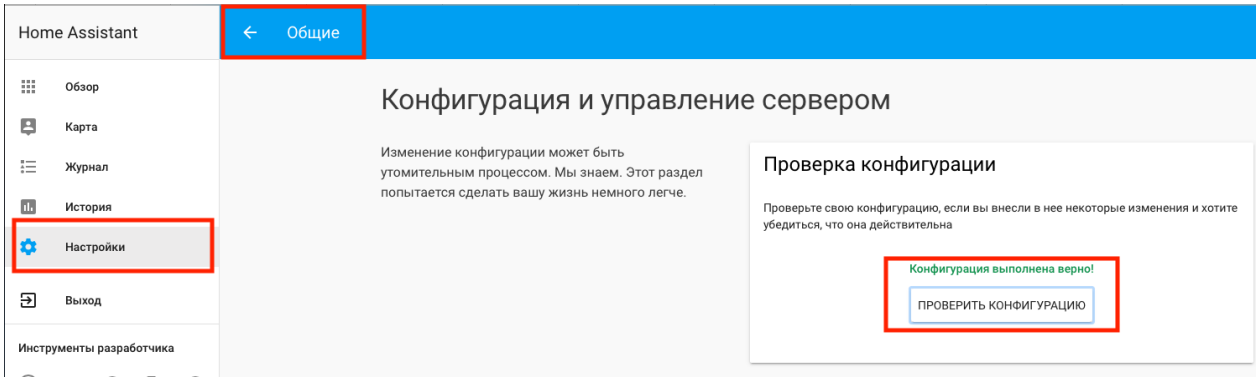

Рис. 4.16 — Налагодження сервера

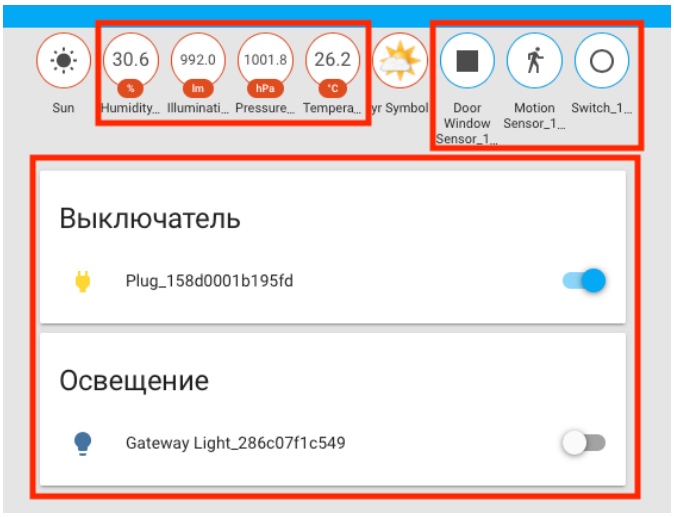

Рис. 4.17 — Підключені пристрої до HomeAssistant

ВИСНОВКИ

У дипломній роботі було розроблено та реалізовано побудова агенту управління та моніторингу «Розумний будинок». В ході розробки було виконано наступні завдання: досліджено основні поняття системи «Розумний будинок»; досліджено існуючі програмні рішення побудови систем «Розумний будинок»; доведено актуальніст у побудові системи управління «Розумний будинок»; описано критерій, що визначатиме оптимальність способу організації системи управління «Розумний будинок»; розроблено програмний продукт, що реалізовує композитний метод побудови системи управління «Розумний будинок»; проведено порівняння результатів побудови системи управління з існуючими рішеннями, розроблено проект на основі описаної в роботі розробки.

1. Ринок наповнений дорогими рішеннями систем управління житловими приміщеннями, не універсальними, обмеженими функціонально. Включають керування багатьма мультимедійними та IT-сервісами, які швидко розвиваються. Таким чином, ми знайшли оптимальні с точки зору функціональності, масштабування, тимчасових, енергетичних, вартісних, масогабаритних та ін. параметри рішень для управління «умним домом». Після всебічного огляду й аналізу існуючих розробок подібних систем прийнято рішення розробляти його як «систему-на-кристалі» (System-on-a-chip, SOC).

2. Апаратна частина агента керування розумним будинком була розроблена на базі ПЛІС-плати (програмована логічна інтегральна схема, FPGA) розробника. Її основу складає ПЛІС Intel Altera, в якій реалізовано SOC NIOS II, - софт-процесор. Для проектування використовується САПР Intel Altera Quartus II та SOPC Builder. Отримані техніко-економичні апаратної частини агенту керування розумного будинку дозволяють при необхідності значно розширити без зміни плати функціонал проекту, реалізувати гарантоздатність агенту, використовувати для його живлення - малопотужні, дешеві джерела живлення, використовувати пасивні системи охолодження, а також зручну та компактну комплектацію корпусу.

3. Програмне забезпечення розроблено для софт процесора, в середовищі Altera Eclipse, на базі цього був створен проект SmartHomeSimulator, розроблений в середовищі Microsoft Visual Studio, програма дозволяє працювати на різних типах плат, моделювати та симулювати системи розумного будинку під різноманітні задачі. Також як допоміжна система виского рівня, в налаштуванні розумного будинку, наводиться платформа під назвою «HomeAssistant», вона дозволяє редагувати та масштабувати агент розумного будинку більш детальніше для потреб зручності звичайних користувачів.

Таким чином запропонован перспективний підхід до створення таких агентів керування розумних будинків, що відрізняються оптимальним комплексом параметрів швидкодії, функціональності, масштабування, енергетичних, вартісних, масогабаритних та ін. для систем керування «розумним будинком». Слід зауважити, що при чисто апаратної реалізації подібного агента на базі ПЛІС, на відміну від чисто програмних рішень, його можна застосовувати і для систем жорсткого реального часу, наприклад, систем моніторингу та керування АЕС.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гололобов,В.Н.«Розумний будинок» своїми руками /В.Н.Гололобов. НТ Пресс, 2007. – 416 с.

2. Платан, А.С. «Електронні компоненти» Каталог, 2008 – 223 с.

3. Тесля, Е.В. «Розумний будинок» своїми руками. будуємо інтелектуальну цифрову систему в своїй квартирі (+ CD), 2008

4. «Розумний будинок HomeAssistant – відкрита платформа домашньої авто матизації ». URL: https://sprut.ai/client/article/155

5. «Системи розумного будинка». URL: https://www.homeassistant.io/

6. Intel FPGA, «Intel Quartus Prime Standard Edition User Guide: Design Compilation», 2019.

7. Intel FPGA, Nios II Gen2 Processor Reference Guide, 2016.

8. Intel FPGA, «Nios II Processor Reference Handbook», 2016.

9. Intel FPGA, «Avalon Interface Specifications», 2018.

10. Intel FPGA, «Nios II Gen2 Software Developer's Handbook», 2018.

11. Д. Харрис и С. Харрис, Цифровая схемотехника и архитектура компьютера, 2017.

12. Смирнов Д.С., Дейнека І.Г., Алейник А.С., Шарков І.А. «Основи розробки вбудованих систем на ПЛІС за використанням процесора Nios II», 2019

13. http://www.aptech.ru/istoriya-razvitiya-sistemy-umnyj-dom

14. https://www.art-in.ru/istoriya-umnogo-doma/

15. https://house-o-matic.com/home-automation-basics/makes-smart-home

16. Jen King, ―Energy Impacts of Smart Home Technologies,‖ April 2018, American Council for an Energy-Efficient Economy, 14.

17. National Institute of Standards and Technology, ―NIST Smart Grid and CPSNewsletter,‖ March 2017. Retrieved July 13, 2018, from https://www.nist. gov/engineering-laboratory/smart-grid/nist-smart-grid-and-cps-newslettermarch-2017.

18. Essie Snell, ―Smart Home Pilots and Programs: A Catalog of Current and

Recent Utility Initiatives,‖ February 2018, E Source, 3.

19. https://www.intel.com/content/www/us/en/programmable/solutions/partners/partner-profile/alorium-technology--llc/board/xlr8-max-10-development-board.html

20. https://www.terasic.com.tw/cgi- bin/page/archive.pl?Language=English&No=1081

Додаток А

Агент системи моніторингу та управління «умним домом».

Специфікація

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ62141_20Б

Аркушів 1

Київ – 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ62141_20Б_81-1	Нікітін_О_С_ТМ62.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ62141_20Б_12-1	Program.cs	Головний модуль
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ62141_20Б_12-2	CommClient.cs	Модуль клієнтської частини
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ62141_20Б_12-3	CommServer.cs	Модуль серверної частини
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ62141_20Б_12-4	ctAirConditioner.cs	Модуль симулювання кондиціонера
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ62141_20Б_12-5	ctBulb.Designer.cs	Модуль симулювання світла
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ62141_20Б_12-6	ctCDplayer.cs	Модуль симулювання CD-програвача
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ62141_20Б_12-7	ctlWindow.cs	Модуль симулювання дверей і окон
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ62141_20Б_12-8	Building.cs	Модуль симулювання будинку
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ62141_20Б_12-9	my_sopc.vhd	Модуль апаратної частини агенту
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ62141_20Б_13-1	Опис.docx	Опис існуючих класів і методів

Додаток Б

Агент системи моніторингу та управління «умним домом»

Лістинг програми

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ62141_20Б_12

Аркушів 24

Київ – 2020

Program.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using SmartHomeSimulator.Model;
using SmartHomeSimulator.Parsing;
using SmartHomeSimulator.Comm;

namespace SmartHomeSimulator
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);

            using (ProtocolManager manager = ProtocolManager.Instance)
            {
                Building building = ModelController.Instance.Model;
                ViewForm view = FormFactory.CreateForm(building);

                Application.ThreadException += Application_ThreadException;
                Application.Run(view);
            }
        }

        static void Application_ThreadException(object sender,
System.Threading.ThreadExceptionEventArgs e)
        {
            ViewForm.Instance.AddLog("Outside");
        }
    }
}
```

CommClient.cs

```
using System.Threading;
using System.Net.Sockets;
using SmartHomeSimulator.Util;
using SmartHomeSimulator.Log;

namespace SmartHomeSimulator.Comm
{
    public class CommClient
    {
        private TcpClient tcpClient;
        private bool keepRunning = false;
        private Thread readingThread = null;
        NetworkStream stream = null;

        public event ClientStoppedEventHandler ClientStopped;

        public CommClient(TcpClient tcpClient)
        {
            this.tcpClient = tcpClient;
            stream = tcpClient.GetStream();
```

```
            readingThread = new Thread(new ThreadStart(run));
            readingThread.Start();
        }

        private void run()
        {
            keepRunning = true;

            while (keepRunning)
            {
                Thread.Sleep(Constants.ShortWait);

                int length = tcpClient.Available;
                if (length > 0)
                {
                    byte[] buffer = new byte[length];
                    stream.Read(buffer, 0, length);

                    string strPacket = Utility.GetString(buffer);

                    handlePacket(strPacket);
                }

                keepRunning = !ProactiveDisconnected;
            }

            keepRunning = false;

            //notify local server
            if (ClientStopped != null)
                ClientStopped(this);
        }

        private void handlePacket(string strPacket)
        {
            try
            {
                LogManager.LogMessage(string.Format("Incomming packet : {0}", strPacket));

                ProtocolPacket packet = ProtocolPacket.Parse(strPacket);
                ProtocolPacket result = ProtocolManager.Instance.ProcessRequestPacket(packet);

                string strResponse = result.ToString();
                if (stream.CanWrite)
                {
                    byte[] buffer = Utility.GetBytes(strResponse);
                    stream.Write(buffer, 0, buffer.Length);
                    stream.Flush();
                }

                LogManager.LogMessage(string.Format("Outgoing packet : {0}", strResponse));
            }
            catch
            {
                LogManager.LogMessage("Result: An error was encountered while handling the
packet.");
            }
        }

        public bool ProactiveDisconnected
        {
            get
            {
                try
```

```
                {
                    if (!tcpClient.Connected)
                        return true;
                    // Detect if client disconnected
                    if (tcpClient.Client.Poll(0, SelectMode.SelectRead))
                    {
                        byte[] buff = new byte[1];
                        if (tcpClient.Client.Receive(buff, SocketFlags.Peek) == 0)
                        {
                            // Client disconnected
                            return true;
                        }
                    }
                }
                catch
                {
                    return true;
                }

                return false;
            }
        }

        internal void Stop()
        {
            if (!keepRunning)
                return;

            this.keepRunning = false;

            if (readingThread != null)
                readingThread.Abort();
        }
    }
}
```

CommServer.cs

```
using System.Collections.Generic;
using System.Net;
using System.Net.Sockets;
using System.Threading;

namespace SmartHomeSimulator.Comm
{
    public class CommServer
    {
        private TcpListener listenerSocket;
        private bool started = false;
        private Thread listeningThread = null;
        private List<CommClient> clients = new List<CommClient>();
        public void Start()
        {
            if (started)
            {
                Stop();
            }

            listeningThread = new Thread(new ThreadStart(run));
            listeningThread.Start();
        }

        private void run()
        {
            listenerSocket = new TcpListener(IPAddress.Any, Constants.ListeningPort);
```

```
            started = true;

            listenerSocket.Start();

            try
            {
                while (started)
                {
                    Thread.Sleep(Constants.LongWait);
                    if (listenerSocket == null)
                        break;

                    if (listenerSocket != null && listenerSocket.Pending())
                    {
                        lock (clients)
                        {
                            CommClient newClient = new
CommClient(listenerSocket.AcceptTcpClient());
                            newClient.ClientStopped += newClient_ClientStopped;
                            this.clients.Add(newClient);
                        }
                    }
                }
            }
            catch { Stop(); }
        }

        void newClient_ClientStopped(CommClient source)
        {
            lock (clients)
            {
                clients.Remove(source);
            }
        }

        public void Stop()
        {
            if (!started)
                return;

            started = false;

            if (listenerSocket != null)
                listenerSocket.Stop();

            if (listeningThread != null)
                listeningThread.Abort();

            lock (clients)
            {
                foreach (CommClient client in clients)
                {
                    client.Stop();
                }

                clients.Clear();
            }
        }

        public bool HasClients
        {
            get
            {
```

```
                lock (clients)
                {
                    return (clients.Count > 0);
                }
            }
        }
    }
}
```

ctAirConditioner.cs

```
namespace SmartHomeSimulator.Controls
{
    partial class ctlAirConditioner
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Component Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.buttonUp = new System.Windows.Forms.Button();
            this.imageList = new System.Windows.Forms.ImageList(this.components);
            this.tableLayoutPanel1 = new System.Windows.Forms.TableLayoutPanel();
            this.panel1 = new System.Windows.Forms.Panel();
            this.labelFunction = new System.Windows.Forms.Label();
            this.labelCurrent = new System.Windows.Forms.Label();
            this.labelDesired = new System.Windows.Forms.Label();
            this.buttonDown = new System.Windows.Forms.Button();
            this.labelID = new System.Windows.Forms.Label();
            this.checkBoxOnlineStatus = new System.Windows.Forms.CheckBox();
            this.checkBoxPoweredOn = new System.Windows.Forms.CheckBox();
            this.timer = new System.Windows.Forms.Timer(this.components);
            this.label1 = new System.Windows.Forms.Label();
            this.tableLayoutPanel1.SuspendLayout();
            this.panel1.SuspendLayout();
            this.SuspendLayout();
            //
            // buttonUp
            //
            this.buttonUp.Anchor = System.Windows.Forms.AnchorStyles.Left;
            this.buttonUp.Location = new System.Drawing.Point(4, 30);
            this.buttonUp.Margin = new System.Windows.Forms.Padding(4, 4, 4, 4);
```

```
            this.buttonUp.Name = "buttonUp";
            this.buttonUp.Size = new System.Drawing.Size(73, 68);
            this.buttonUp.TabIndex = 1;
            this.buttonUp.UseVisualStyleBackColor = true;
            this.buttonUp.Click += new System.EventHandler(this.buttonUp_Click);
            //
            // imageList
            //
            this.imageList.ColorDepth = System.Windows.Forms.ColorDepth.Depth24Bit;
            this.imageList.ImageSize = new System.Drawing.Size(48, 48);
            this.imageList.TransparentColor = System.Drawing.Color.Transparent;
            //
            // tableLayoutPanel1
            //
            this.tableLayoutPanel1.ColumnCount = 3;
            this.tableLayoutPanel1.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 50F));
            this.tableLayoutPanel1.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Absolute, 333F));
            this.tableLayoutPanel1.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 50F));
            this.tableLayoutPanel1.Controls.Add(this.panel1, 1, 0);
            this.tableLayoutPanel1.Dock = System.Windows.Forms.DockStyle.Top;
            this.tableLayoutPanel1.Location = new System.Drawing.Point(0, 0);
            this.tableLayoutPanel1.Margin = new System.Windows.Forms.Padding(4, 4, 4, 4);
            this.tableLayoutPanel1.Name = "tableLayoutPanel1";
            this.tableLayoutPanel1.RowCount = 1;
            this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 100F));
            this.tableLayoutPanel1.Size = new System.Drawing.Size(359, 127);
            this.tableLayoutPanel1.TabIndex = 3;
            //
            // panel1
            //
            this.panel1.Controls.Add(this.labelFunction);
            this.panel1.Controls.Add(this.labelCurrent);
            this.panel1.Controls.Add(this.labelDesired);
            this.panel1.Controls.Add(this.buttonDown);
            this.panel1.Controls.Add(this.buttonUp);
            this.panel1.Dock = System.Windows.Forms.DockStyle.Fill;
            this.panel1.Location = new System.Drawing.Point(17, 4);
            this.panel1.Margin = new System.Windows.Forms.Padding(4, 4, 4, 4);
            this.panel1.Name = "panel1";
            this.panel1.Size = new System.Drawing.Size(325, 119);
            this.panel1.TabIndex = 4;
            //
            // labelFunction
            //
            this.labelFunction.Anchor = System.Windows.Forms.AnchorStyles.None;
            this.labelFunction.AutoSize = true;
            this.labelFunction.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.labelFunction.Location = new System.Drawing.Point(111, 90);
            this.labelFunction.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);
            this.labelFunction.Name = "labelFunction";
            this.labelFunction.Size = new System.Drawing.Size(65, 20);
            this.labelFunction.TabIndex = 6;
            this.labelFunction.Text = "Cooling";
            this.labelFunction.TextAlign = System.Drawing.ContentAlignment.BottomLeft;
            //
            // labelCurrent
            //
            this.labelCurrent.Anchor = System.Windows.Forms.AnchorStyles.None;
            this.labelCurrent.AutoSize = true;
```

```
            this.labelCurrent.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.labelCurrent.Location = new System.Drawing.Point(111, 7);
            this.labelCurrent.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);
            this.labelCurrent.Name = "labelCurrent";
            this.labelCurrent.Size = new System.Drawing.Size(51, 20);
            this.labelCurrent.TabIndex = 5;
            this.labelCurrent.Text = "30 °C";
            //
            // labelDesired
            //
            this.labelDesired.Anchor = System.Windows.Forms.AnchorStyles.None;
            this.labelDesired.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
            this.labelDesired.Font = new System.Drawing.Font("Microsoft Sans Serif", 20F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.labelDesired.Location = new System.Drawing.Point(107, 4);
            this.labelDesired.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);
            this.labelDesired.Name = "labelDesired";
            this.labelDesired.Size = new System.Drawing.Size(113, 110);
            this.labelDesired.TabIndex = 4;
            this.labelDesired.Text = "22";
            this.labelDesired.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
            //
            // buttonDown
            //
            this.buttonDown.Anchor = System.Windows.Forms.AnchorStyles.Right;
            this.buttonDown.Location = new System.Drawing.Point(248, 30);
            this.buttonDown.Margin = new System.Windows.Forms.Padding(4, 4, 4, 4);
            this.buttonDown.Name = "buttonDown";
            this.buttonDown.Size = new System.Drawing.Size(73, 68);
            this.buttonDown.TabIndex = 3;
            this.buttonDown.UseVisualStyleBackColor = true;
            this.buttonDown.Click += new System.EventHandler(this.buttonDown_Click);
            //
            // labelID
            //
            this.labelID.Anchor                                                    =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom                |
System.Windows.Forms.AnchorStyles.Right)));
            this.labelID.AutoSize = true;
            this.labelID.Location = new System.Drawing.Point(331, 132);
            this.labelID.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);
            this.labelID.Name = "labelID";
            this.labelID.Size = new System.Drawing.Size(21, 17);
            this.labelID.TabIndex = 10;
            this.labelID.Text = "ID";
            //
            // checkBoxOnlineStatus
            //
            this.checkBoxOnlineStatus.AutoSize = true;
            this.checkBoxOnlineStatus.Checked = true;
            this.checkBoxOnlineStatus.CheckState = System.Windows.Forms.CheckState.Checked;
            this.checkBoxOnlineStatus.Location = new System.Drawing.Point(91, 130);
            this.checkBoxOnlineStatus.Margin = new System.Windows.Forms.Padding(4, 4, 4, 4);
            this.checkBoxOnlineStatus.Name = "checkBoxOnlineStatus";
            this.checkBoxOnlineStatus.Size = new System.Drawing.Size(71, 21);
            this.checkBoxOnlineStatus.TabIndex = 9;
            this.checkBoxOnlineStatus.Text = "Online";
            this.checkBoxOnlineStatus.UseVisualStyleBackColor = true;
            this.checkBoxOnlineStatus.CheckedChanged                    +=                    new
System.EventHandler(this.checkBoxOnlineStatus_CheckedChanged);
            //
            // checkBoxPoweredOn
            //
```

```
            this.checkBoxPoweredOn.AutoSize = true;
            this.checkBoxPoweredOn.Location = new System.Drawing.Point(4, 130);
            this.checkBoxPoweredOn.Margin = new System.Windows.Forms.Padding(4, 4, 4, 4);
            this.checkBoxPoweredOn.Name = "checkBoxPoweredOn";
            this.checkBoxPoweredOn.Size = new System.Drawing.Size(69, 21);
            this.checkBoxPoweredOn.TabIndex = 8;
            this.checkBoxPoweredOn.Text = "Power";
            this.checkBoxPoweredOn.UseVisualStyleBackColor = true;
            this.checkBoxPoweredOn.CheckedChanged                    +=                    new
System.EventHandler(this.checkBoxPoweredOn_CheckedChanged);
            //
            // timer
            //
            this.timer.Interval = 1000;
            this.timer.Tick += new System.EventHandler(this.timer_Tick);
            //
            // label1
            //
            this.label1.Anchor                                                                =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom                |
System.Windows.Forms.AnchorStyles.Right)));
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(227, 131);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(101, 17);
            this.label1.TabIndex = 11;
            this.label1.Text = "Air Conditioner";
            //
            // ctlAirConditioner
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 16F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
            this.Controls.Add(this.label1);
            this.Controls.Add(this.labelID);
            this.Controls.Add(this.checkBoxOnlineStatus);
            this.Controls.Add(this.checkBoxPoweredOn);
            this.Controls.Add(this.tableLayoutPanel1);
            this.Margin = new System.Windows.Forms.Padding(4, 4, 4, 4);
            this.Name = "ctlAirConditioner";
            this.Size = new System.Drawing.Size(359, 154);
            this.tableLayoutPanel1.ResumeLayout(false);
            this.panel1.ResumeLayout(false);
            this.panel1.PerformLayout();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Button buttonUp;
        private System.Windows.Forms.ImageList imageList;
        private System.Windows.Forms.TableLayoutPanel tableLayoutPanel1;
        private System.Windows.Forms.Button buttonDown;
        private System.Windows.Forms.Panel panel1;
        private System.Windows.Forms.Label labelID;
        private System.Windows.Forms.CheckBox checkBoxOnlineStatus;
        private System.Windows.Forms.CheckBox checkBoxPoweredOn;
        private System.Windows.Forms.Label labelFunction;
        private System.Windows.Forms.Label labelCurrent;
        private System.Windows.Forms.Label labelDesired;
        private System.Windows.Forms.Timer timer;
        private System.Windows.Forms.Label label1;
```

```
    }
}
```

ctBulb.Designer.cs

```
namespace SmartHomeSimulator.Controls
{
    partial class ctlBulb
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Component Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.imageList = new System.Windows.Forms.ImageList(this.components);
            this.checkBoxPoweredOn = new System.Windows.Forms.CheckBox();
            this.checkBoxOnlineStatus = new System.Windows.Forms.CheckBox();
            this.pictureBox = new System.Windows.Forms.PictureBox();
            this.labelID = new System.Windows.Forms.Label();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox)).BeginInit();
            this.SuspendLayout();
            //
            // imageList
            //
            this.imageList.ColorDepth = System.Windows.Forms.ColorDepth.Depth24Bit;
            this.imageList.ImageSize = new System.Drawing.Size(60, 90);
            this.imageList.TransparentColor = System.Drawing.Color.Transparent;
            //
            // checkBoxPoweredOn
            //
            this.checkBoxPoweredOn.AutoSize = true;
            this.checkBoxPoweredOn.Location = new System.Drawing.Point(3, 103);
            this.checkBoxPoweredOn.Name = "checkBoxPoweredOn";
            this.checkBoxPoweredOn.Size = new System.Drawing.Size(56, 17);
            this.checkBoxPoweredOn.TabIndex = 1;
            this.checkBoxPoweredOn.Text = "Power";
            this.checkBoxPoweredOn.UseVisualStyleBackColor = true;
            this.checkBoxPoweredOn.CheckedChanged += new
System.EventHandler(this.checkBoxPoweredOn_CheckedChanged);
            //
            // checkBoxOnlineStatus
            //
```

```
            this.checkBoxOnlineStatus.AutoSize = true;
            this.checkBoxOnlineStatus.Checked = true;
            this.checkBoxOnlineStatus.CheckState = System.Windows.Forms.CheckState.Checked;
            this.checkBoxOnlineStatus.Location = new System.Drawing.Point(68, 103);
            this.checkBoxOnlineStatus.Name = "checkBoxOnlineStatus";
            this.checkBoxOnlineStatus.Size = new System.Drawing.Size(56, 17);
            this.checkBoxOnlineStatus.TabIndex = 2;
            this.checkBoxOnlineStatus.Text = "Online";
            this.checkBoxOnlineStatus.UseVisualStyleBackColor = true;
            this.checkBoxOnlineStatus.CheckedChanged += new
System.EventHandler(this.checkBoxOnlineStatus_CheckedChanged);
            //
            // pictureBox
            //
            this.pictureBox.Dock = System.Windows.Forms.DockStyle.Top;
            this.pictureBox.Location = new System.Drawing.Point(0, 0);
            this.pictureBox.Name = "pictureBox";
            this.pictureBox.Size = new System.Drawing.Size(165, 97);
            this.pictureBox.SizeMode = System.Windows.Forms.PictureBoxSizeMode.CenterImage;
            this.pictureBox.TabIndex = 0;
            this.pictureBox.TabStop = false;
            //
            // labelID
            //
            this.labelID.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Right)));
            this.labelID.AutoSize = true;
            this.labelID.Location = new System.Drawing.Point(144, 103);
            this.labelID.Name = "labelID";
            this.labelID.Size = new System.Drawing.Size(18, 13);
            this.labelID.TabIndex = 3;
            this.labelID.Text = "ID";
            //
            // ctlBulb
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
            this.Controls.Add(this.labelID);
            this.Controls.Add(this.checkBoxOnlineStatus);
            this.Controls.Add(this.checkBoxPoweredOn);
            this.Controls.Add(this.pictureBox);
            this.MinimumSize = new System.Drawing.Size(125, 125);
            this.Name = "ctlBulb";
            this.Size = new System.Drawing.Size(165, 123);
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.ImageList imageList;
        private System.Windows.Forms.PictureBox pictureBox;
        private System.Windows.Forms.CheckBox checkBoxPoweredOn;
        private System.Windows.Forms.CheckBox checkBoxOnlineStatus;
        private System.Windows.Forms.Label labelID;
    }
}
```

ctlCDPlayer.cs

```
namespace SmartHomeSimulator.Controls
{
```

```
    partial class ctlCDPlayer
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Component Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.imageList = new System.Windows.Forms.ImageList(this.components);
            this.labelStatus = new System.Windows.Forms.Label();
            this.checkBoxOnlineStatus = new System.Windows.Forms.CheckBox();
            this.checkBoxPoweredOn = new System.Windows.Forms.CheckBox();
            this.labelID = new System.Windows.Forms.Label();
            this.pictureBoxPrev = new System.Windows.Forms.PictureBox();
            this.pictureBoxNext = new System.Windows.Forms.PictureBox();
            this.pictureBoxPlay = new System.Windows.Forms.PictureBox();
            this.pictureBoxStop = new System.Windows.Forms.PictureBox();
            this.pictureBoxPause = new System.Windows.Forms.PictureBox();
            this.label1 = new System.Windows.Forms.Label();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBoxPrev)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBoxNext)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBoxPlay)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBoxStop)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBoxPause)).BeginInit();
            this.SuspendLayout();
            //
            // imageList
            //
            this.imageList.ColorDepth = System.Windows.Forms.ColorDepth.Depth24Bit;
            this.imageList.ImageSize = new System.Drawing.Size(57, 57);
            this.imageList.TransparentColor = System.Drawing.Color.Transparent;
            //
            // labelStatus
            //
            this.labelStatus.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
            this.labelStatus.Dock = System.Windows.Forms.DockStyle.Top;
            this.labelStatus.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
            this.labelStatus.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.labelStatus.Location = new System.Drawing.Point(0, 0);
            this.labelStatus.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);
            this.labelStatus.Name = "labelStatus";
```

```
            this.labelStatus.Size = new System.Drawing.Size(464, 35);
            this.labelStatus.TabIndex = 0;
            this.labelStatus.Text = "Powered Off";
            this.labelStatus.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
            //
            // checkBoxOnlineStatus
            //
            this.checkBoxOnlineStatus.AutoSize = true;
            this.checkBoxOnlineStatus.Checked = true;
            this.checkBoxOnlineStatus.CheckState = System.Windows.Forms.CheckState.Checked;
            this.checkBoxOnlineStatus.Location = new System.Drawing.Point(87, 127);
            this.checkBoxOnlineStatus.Margin = new System.Windows.Forms.Padding(4, 4, 4, 4);
            this.checkBoxOnlineStatus.Name = "checkBoxOnlineStatus";
            this.checkBoxOnlineStatus.Size = new System.Drawing.Size(71, 21);
            this.checkBoxOnlineStatus.TabIndex = 8;
            this.checkBoxOnlineStatus.Text = "Online";
            this.checkBoxOnlineStatus.UseVisualStyleBackColor = true;
            this.checkBoxOnlineStatus.CheckedChanged                    +=                    new
System.EventHandler(this.checkBoxOnlineStatus_CheckedChanged);
            //
            // checkBoxPoweredOn
            //
            this.checkBoxPoweredOn.AutoSize = true;
            this.checkBoxPoweredOn.Location = new System.Drawing.Point(4, 127);
            this.checkBoxPoweredOn.Margin = new System.Windows.Forms.Padding(4, 4, 4, 4);
            this.checkBoxPoweredOn.Name = "checkBoxPoweredOn";
            this.checkBoxPoweredOn.Size = new System.Drawing.Size(69, 21);
            this.checkBoxPoweredOn.TabIndex = 7;
            this.checkBoxPoweredOn.Text = "Power";
            this.checkBoxPoweredOn.UseVisualStyleBackColor = true;
            this.checkBoxPoweredOn.CheckedChanged                    +=                    new
System.EventHandler(this.checkBoxPoweredOn_CheckedChanged);
            //
            // labelID
            //
            this.labelID.Anchor                                                                    =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom                 |
System.Windows.Forms.AnchorStyles.Right)));
            this.labelID.AutoSize = true;
            this.labelID.Location = new System.Drawing.Point(437, 128);
            this.labelID.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);
            this.labelID.Name = "labelID";
            this.labelID.Size = new System.Drawing.Size(21, 17);
            this.labelID.TabIndex = 10;
            this.labelID.Text = "ID";
            //
            // pictureBoxPrev
            //
            this.pictureBoxPrev.Anchor = System.Windows.Forms.AnchorStyles.None;
            this.pictureBoxPrev.Location = new System.Drawing.Point(25, 43);
            this.pictureBoxPrev.Margin = new System.Windows.Forms.Padding(4, 4, 4, 4);
            this.pictureBoxPrev.Name = "pictureBoxPrev";
            this.pictureBoxPrev.Size = new System.Drawing.Size(76, 70);
            this.pictureBoxPrev.SizeMode = System.Windows.Forms.PictureBoxSizeMode.CenterImage;
            this.pictureBoxPrev.TabIndex = 16;
            this.pictureBoxPrev.TabStop = false;
            this.pictureBoxPrev.Click += new System.EventHandler(this.pictureBoxPrev_Click);
            //
            // pictureBoxNext
            //
            this.pictureBoxNext.Anchor = System.Windows.Forms.AnchorStyles.None;
            this.pictureBoxNext.Location = new System.Drawing.Point(109, 43);
            this.pictureBoxNext.Margin = new System.Windows.Forms.Padding(4, 4, 4, 4);
            this.pictureBoxNext.Name = "pictureBoxNext";
```

```
            this.pictureBoxNext.Size = new System.Drawing.Size(76, 70);
            this.pictureBoxNext.TabIndex = 15;
            this.pictureBoxNext.TabStop = false;
            this.pictureBoxNext.Click += new System.EventHandler(this.pictureBoxNext_Click);
            //
            // pictureBoxPlay
            //
            this.pictureBoxPlay.Anchor = System.Windows.Forms.AnchorStyles.None;
            this.pictureBoxPlay.Location = new System.Drawing.Point(193, 43);
            this.pictureBoxPlay.Margin = new System.Windows.Forms.Padding(4, 4, 4, 4);
            this.pictureBoxPlay.Name = "pictureBoxPlay";
            this.pictureBoxPlay.Size = new System.Drawing.Size(76, 70);
            this.pictureBoxPlay.TabIndex = 14;
            this.pictureBoxPlay.TabStop = false;
            this.pictureBoxPlay.Click += new System.EventHandler(this.pictureBoxPlay_Click);
            //
            // pictureBoxStop
            //
            this.pictureBoxStop.Anchor = System.Windows.Forms.AnchorStyles.None;
            this.pictureBoxStop.Location = new System.Drawing.Point(277, 43);
            this.pictureBoxStop.Margin = new System.Windows.Forms.Padding(4, 4, 4, 4);
            this.pictureBoxStop.Name = "pictureBoxStop";
            this.pictureBoxStop.Size = new System.Drawing.Size(76, 70);
            this.pictureBoxStop.TabIndex = 13;
            this.pictureBoxStop.TabStop = false;
            this.pictureBoxStop.Click += new System.EventHandler(this.pictureBoxStop_Click);
            //
            // pictureBoxPause
            //
            this.pictureBoxPause.Anchor = System.Windows.Forms.AnchorStyles.None;
            this.pictureBoxPause.Location = new System.Drawing.Point(361, 43);
            this.pictureBoxPause.Margin = new System.Windows.Forms.Padding(4, 4, 4, 4);
            this.pictureBoxPause.Name = "pictureBoxPause";
            this.pictureBoxPause.Size = new System.Drawing.Size(76, 70);
            this.pictureBoxPause.TabIndex = 12;
            this.pictureBoxPause.TabStop = false;
            this.pictureBoxPause.Click += new System.EventHandler(this.pictureBoxPause_Click);
            //
            // label1
            //
            this.label1.Anchor                                                                  =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom                 |
System.Windows.Forms.AnchorStyles.Right)));
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(347, 127);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(90, 17);
            this.label1.TabIndex = 17;
            this.label1.Text = "Media Player";
            //
            // ctlCDPlayer
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 16F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
            this.Controls.Add(this.label1);
            this.Controls.Add(this.pictureBoxPrev);
            this.Controls.Add(this.pictureBoxNext);
            this.Controls.Add(this.pictureBoxPlay);
            this.Controls.Add(this.pictureBoxStop);
            this.Controls.Add(this.pictureBoxPause);
            this.Controls.Add(this.labelID);
            this.Controls.Add(this.checkBoxOnlineStatus);
            this.Controls.Add(this.checkBoxPoweredOn);
```

```
            this.Controls.Add(this.labelStatus);
            this.Margin = new System.Windows.Forms.Padding(4, 4, 4, 4);
            this.MinimumSize = new System.Drawing.Size(466, 153);
            this.Name = "ctlCDPlayer";
            this.Size = new System.Drawing.Size(464, 151);
            ((System.ComponentModel.ISupportInitialize)(this.pictureBoxPrev)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBoxNext)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBoxPlay)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBoxStop)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBoxPause)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.ImageList imageList;
        private System.Windows.Forms.Label labelStatus;
        private System.Windows.Forms.CheckBox checkBoxOnlineStatus;
        private System.Windows.Forms.CheckBox checkBoxPoweredOn;
        private System.Windows.Forms.Label labelID;
        private System.Windows.Forms.PictureBox pictureBoxPrev;
        private System.Windows.Forms.PictureBox pictureBoxNext;
        private System.Windows.Forms.PictureBox pictureBoxPlay;
        private System.Windows.Forms.PictureBox pictureBoxStop;
        private System.Windows.Forms.PictureBox pictureBoxPause;
        private System.Windows.Forms.Label label1;
    }
}
```

ctlWindow.cs

```
namespace SmartHomeSimulator.Controls
{
    partial class ctlWindow
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Component Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
```

```
            this.imageList = new System.Windows.Forms.ImageList(this.components);
            this.checkBoxOnlineStatus = new System.Windows.Forms.CheckBox();
            this.checkBoxPoweredOn = new System.Windows.Forms.CheckBox();
            this.checkBoxSafe = new System.Windows.Forms.CheckBox();
            this.tableLayoutPanel1 = new System.Windows.Forms.TableLayoutPanel();
            this.labelID = new System.Windows.Forms.Label();
            this.pictureBox = new System.Windows.Forms.PictureBox();
            this.checkBoxOpen = new System.Windows.Forms.CheckBox();
            this.tableLayoutPanel1.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox)).BeginInit();
            this.SuspendLayout();
            //
            // imageList
            //
            this.imageList.ColorDepth = System.Windows.Forms.ColorDepth.Depth24Bit;
            this.imageList.ImageSize = new System.Drawing.Size(256, 256);
            this.imageList.TransparentColor = System.Drawing.Color.Transparent;
            //
            // checkBoxOnlineStatus
            //
            this.checkBoxOnlineStatus.AutoSize = true;
            this.checkBoxOnlineStatus.Checked = true;
            this.checkBoxOnlineStatus.CheckState = System.Windows.Forms.CheckState.Checked;
            this.checkBoxOnlineStatus.Location = new System.Drawing.Point(65, 228);
            this.checkBoxOnlineStatus.Name = "checkBoxOnlineStatus";
            this.checkBoxOnlineStatus.Size = new System.Drawing.Size(56, 17);
            this.checkBoxOnlineStatus.TabIndex = 4;
            this.checkBoxOnlineStatus.Text = "Online";
            this.checkBoxOnlineStatus.UseVisualStyleBackColor = true;
            this.checkBoxOnlineStatus.CheckedChanged += new
System.EventHandler(this.checkBoxOnlineStatus_CheckedChanged);
            //
            // checkBoxPoweredOn
            //
            this.checkBoxPoweredOn.AutoSize = true;
            this.checkBoxPoweredOn.Location = new System.Drawing.Point(3, 228);
            this.checkBoxPoweredOn.Name = "checkBoxPoweredOn";
            this.checkBoxPoweredOn.Size = new System.Drawing.Size(56, 17);
            this.checkBoxPoweredOn.TabIndex = 3;
            this.checkBoxPoweredOn.Text = "Power";
            this.checkBoxPoweredOn.UseVisualStyleBackColor = true;
            this.checkBoxPoweredOn.CheckedChanged += new
System.EventHandler(this.checkBoxPoweredOn_CheckedChanged);
            //
            // checkBoxSafe
            //
            this.checkBoxSafe.AutoSize = true;
            this.checkBoxSafe.Checked = true;
            this.checkBoxSafe.CheckState = System.Windows.Forms.CheckState.Checked;
            this.checkBoxSafe.Location = new System.Drawing.Point(127, 228);
            this.checkBoxSafe.Name = "checkBoxSafe";
            this.checkBoxSafe.Size = new System.Drawing.Size(48, 17);
            this.checkBoxSafe.TabIndex = 5;
            this.checkBoxSafe.Text = "Safe";
            this.checkBoxSafe.UseVisualStyleBackColor = true;
            this.checkBoxSafe.CheckedChanged += new
System.EventHandler(this.checkBoxSafe_CheckedChanged);
            //
            // tableLayoutPanel1
            //
            this.tableLayoutPanel1.ColumnCount = 3;
            this.tableLayoutPanel1.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 50F));
            this.tableLayoutPanel1.ColumnStyles.Add(new
```

```
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Absolute, 206F));
            this.tableLayoutPanel1.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 50F));
            this.tableLayoutPanel1.Controls.Add(this.pictureBox, 1, 0);
            this.tableLayoutPanel1.Dock = System.Windows.Forms.DockStyle.Top;
            this.tableLayoutPanel1.Location = new System.Drawing.Point(0, 0);
            this.tableLayoutPanel1.Name = "tableLayoutPanel1";
            this.tableLayoutPanel1.RowCount = 1;
            this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 100F));
            this.tableLayoutPanel1.Size = new System.Drawing.Size(260, 209);
            this.tableLayoutPanel1.TabIndex = 6;
            //
            // labelID
            //
            this.labelID.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Right)));
            this.labelID.AutoSize = true;
            this.labelID.Location = new System.Drawing.Point(240, 229);
            this.labelID.Name = "labelID";
            this.labelID.Size = new System.Drawing.Size(18, 13);
            this.labelID.TabIndex = 7;
            this.labelID.Text = "ID";
            //
            // pictureBox
            //
            this.pictureBox.Location = new System.Drawing.Point(30, 3);
            this.pictureBox.MaximumSize = new System.Drawing.Size(256, 256);
            this.pictureBox.Name = "pictureBox";
            this.pictureBox.Size = new System.Drawing.Size(200, 200);
            this.pictureBox.SizeMode = System.Windows.Forms.PictureBoxSizeMode.Zoom;
            this.pictureBox.TabIndex = 1;
            this.pictureBox.TabStop = false;
            //
            // checkBoxOpen
            //
            this.checkBoxOpen.AutoSize = true;
            this.checkBoxOpen.Location = new System.Drawing.Point(181, 228);
            this.checkBoxOpen.Name = "checkBoxOpen";
            this.checkBoxOpen.Size = new System.Drawing.Size(52, 17);
            this.checkBoxOpen.TabIndex = 8;
            this.checkBoxOpen.Text = "Open";
            this.checkBoxOpen.UseVisualStyleBackColor = true;
            this.checkBoxOpen.CheckedChanged += new
System.EventHandler(this.checkBoxOpen_CheckedChanged);
            //
            // ctlWindow
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
            this.Controls.Add(this.checkBoxOpen);
            this.Controls.Add(this.labelID);
            this.Controls.Add(this.tableLayoutPanel1);
            this.Controls.Add(this.checkBoxSafe);
            this.Controls.Add(this.checkBoxOnlineStatus);
            this.Controls.Add(this.checkBoxPoweredOn);
            this.MinimumSize = new System.Drawing.Size(262, 250);
            this.Name = "ctlWindow";
            this.Size = new System.Drawing.Size(260, 250);
            this.tableLayoutPanel1.ResumeLayout(false);
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox)).EndInit();
            this.ResumeLayout(false);
```

```
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.ImageList imageList;
        private System.Windows.Forms.CheckBox checkBoxOnlineStatus;
        private System.Windows.Forms.CheckBox checkBoxPoweredOn;
        private System.Windows.Forms.CheckBox checkBoxSafe;
        private System.Windows.Forms.TableLayoutPanel tableLayoutPanel1;
        private System.Windows.Forms.PictureBox pictureBox;
        private System.Windows.Forms.Label labelID;
        private System.Windows.Forms.CheckBox checkBoxOpen;
    }
}
```

Building.cs

```
using System.Collections.Generic;
using SmartHomeSimulator.Util;

namespace SmartHomeSimulator.Model
{
    public class Building : MetaModel
    {
        public IList<Floor> Floors { get; set; }

        public string StreetAddress { get; set; }

        public Building()
        {
            Name = "Smart Home v1.0";
            Floors = new SerializeableList<Floor>();
        }
    }
}
```

my_sopc.vhd

```
library altera;
use altera.altera_europa_support_lib.all;

library altera_mf;
use altera_mf.altera_mf_components.all;

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

library std;
use std.textio.all;

entity cpu_0_jtag_debug_module_arbitrator is
        port (
              -- inputs:
                 signal clk : IN STD_LOGIC;
```

```
                signal cpu_0_data_master_address_to_slave : IN STD_LOGIC_VECTOR (13 DOWNTO 0);
                signal cpu_0_data_master_byteenable : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
                signal cpu_0_data_master_debugaccess : IN STD_LOGIC;
                signal cpu_0_data_master_read : IN STD_LOGIC;
                signal cpu_0_data_master_waitrequest : IN STD_LOGIC;
                signal cpu_0_data_master_write : IN STD_LOGIC;
                signal cpu_0_data_master_writedata : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
                signal  cpu_0_instruction_master_address_to_slave  :  IN  STD_LOGIC_VECTOR  (13  DOWNTO
0);
                signal cpu_0_instruction_master_read : IN STD_LOGIC;
                signal cpu_0_jtag_debug_module_readdata : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
                signal cpu_0_jtag_debug_module_resetrequest : IN STD_LOGIC;
                signal reset_n : IN STD_LOGIC;

             -- outputs:
                signal cpu_0_data_master_granted_cpu_0_jtag_debug_module : OUT STD_LOGIC;
                signal cpu_0_data_master_qualified_request_cpu_0_jtag_debug_module : OUT STD_LOGIC;
                signal cpu_0_data_master_read_data_valid_cpu_0_jtag_debug_module : OUT STD_LOGIC;
                signal cpu_0_data_master_requests_cpu_0_jtag_debug_module : OUT STD_LOGIC;
                signal cpu_0_instruction_master_granted_cpu_0_jtag_debug_module : OUT STD_LOGIC;
                signal   cpu_0_instruction_master_qualified_request_cpu_0_jtag_debug_module   :   OUT
STD_LOGIC;
                signal   cpu_0_instruction_master_read_data_valid_cpu_0_jtag_debug_module   :   OUT
STD_LOGIC;
                signal cpu_0_instruction_master_requests_cpu_0_jtag_debug_module : OUT STD_LOGIC;
                signal cpu_0_jtag_debug_module_address : OUT STD_LOGIC_VECTOR (8 DOWNTO 0);
                signal cpu_0_jtag_debug_module_begintransfer : OUT STD_LOGIC;
                signal cpu_0_jtag_debug_module_byteenable : OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
                signal cpu_0_jtag_debug_module_chipselect : OUT STD_LOGIC;
                signal cpu_0_jtag_debug_module_debugaccess : OUT STD_LOGIC;
                signal  cpu_0_jtag_debug_module_readdata_from_sa  :  OUT  STD_LOGIC_VECTOR  (31  DOWNTO
0);
                signal cpu_0_jtag_debug_module_reset_n : OUT STD_LOGIC;
                signal cpu_0_jtag_debug_module_resetrequest_from_sa : OUT STD_LOGIC;
                signal cpu_0_jtag_debug_module_write : OUT STD_LOGIC;
                signal cpu_0_jtag_debug_module_writedata : OUT STD_LOGIC_VECTOR (31 DOWNTO 0);
                signal d1_cpu_0_jtag_debug_module_end_xfer : OUT STD_LOGIC
             );
end entity cpu_0_jtag_debug_module_arbitrator;


architecture europa of cpu_0_jtag_debug_module_arbitrator is
               signal cpu_0_data_master_arbiterlock :  STD_LOGIC;
               signal cpu_0_data_master_arbiterlock2 :  STD_LOGIC;
               signal cpu_0_data_master_continuerequest :  STD_LOGIC;
```

```
                signal cpu_0_data_master_saved_grant_cpu_0_jtag_debug_module :  STD_LOGIC;
                signal cpu_0_instruction_master_arbiterlock :  STD_LOGIC;
                signal cpu_0_instruction_master_arbiterlock2 :  STD_LOGIC;
                signal cpu_0_instruction_master_continuerequest :  STD_LOGIC;
                signal cpu_0_instruction_master_saved_grant_cpu_0_jtag_debug_module :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_allgrants :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_allow_new_arb_cycle :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_any_bursting_master_saved_grant :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_any_continuerequest :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_arb_addend :  STD_LOGIC_VECTOR (1 DOWNTO 0);
                signal cpu_0_jtag_debug_module_arb_counter_enable :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_arb_share_counter :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_arb_share_counter_next_value :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_arb_share_set_values :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_arb_winner :  STD_LOGIC_VECTOR (1 DOWNTO 0);
                signal cpu_0_jtag_debug_module_arbitration_holdoff_internal :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_beginbursttransfer_internal :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_begins_xfer :  STD_LOGIC;
                signal  cpu_0_jtag_debug_module_chosen_master_double_vector  :   STD_LOGIC_VECTOR  (3
DOWNTO 0);
                signal cpu_0_jtag_debug_module_chosen_master_rot_left :  STD_LOGIC_VECTOR (1 DOWNTO
0);
                signal cpu_0_jtag_debug_module_end_xfer :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_firsttransfer :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_grant_vector :  STD_LOGIC_VECTOR (1 DOWNTO 0);
                signal cpu_0_jtag_debug_module_in_a_read_cycle :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_in_a_write_cycle :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_master_qreq_vector :  STD_LOGIC_VECTOR (1 DOWNTO 0);
                signal cpu_0_jtag_debug_module_non_bursting_master_requests :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_reg_firsttransfer :  STD_LOGIC;
                signal  cpu_0_jtag_debug_module_saved_chosen_master_vector  :   STD_LOGIC_VECTOR  (1
DOWNTO 0);
                signal cpu_0_jtag_debug_module_slavearbiterlockenable :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_slavearbiterlockenable2 :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_unreg_firsttransfer :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_waits_for_read :  STD_LOGIC;
                signal cpu_0_jtag_debug_module_waits_for_write :  STD_LOGIC;
                signal d1_reasons_to_wait :  STD_LOGIC;
                signal enable_nonzero_assertions :  STD_LOGIC;
                signal end_xfer_arb_share_counter_term_cpu_0_jtag_debug_module :  STD_LOGIC;
                signal in_a_read_cycle :  STD_LOGIC;
                signal in_a_write_cycle :  STD_LOGIC;
                signal internal_cpu_0_data_master_granted_cpu_0_jtag_debug_module :  STD_LOGIC;
                signal    internal_cpu_0_data_master_qualified_request_cpu_0_jtag_debug_module    :
STD_LOGIC;
```

```
                signal internal_cpu_0_data_master_requests_cpu_0_jtag_debug_module :  STD_LOGIC;
                signal      internal_cpu_0_instruction_master_granted_cpu_0_jtag_debug_module      :
STD_LOGIC;
                signal internal_cpu_0_instruction_master_qualified_request_cpu_0_jtag_debug_module :
STD_LOGIC;
                signal      internal_cpu_0_instruction_master_requests_cpu_0_jtag_debug_module     :
STD_LOGIC;
                signal      last_cycle_cpu_0_data_master_granted_slave_cpu_0_jtag_debug_module     :
STD_LOGIC;
                signal  last_cycle_cpu_0_instruction_master_granted_slave_cpu_0_jtag_debug_module  :
STD_LOGIC;
                signal      shifted_address_to_cpu_0_jtag_debug_module_from_cpu_0_data_master      :
STD_LOGIC_VECTOR (13 DOWNTO 0);
                signal   shifted_address_to_cpu_0_jtag_debug_module_from_cpu_0_instruction_master   :
STD_LOGIC_VECTOR (13 DOWNTO 0);
                signal wait_for_cpu_0_jtag_debug_module_counter :  STD_LOGIC;
```

Додаток В

Агент системи моніторингу та управління «умним домом»

Опис програмного коду

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ62141_20Б_13-1

Аркушів 4

Київ – 2020

Ініціалізація та розпорядження :

Ініціалізація SDK по суті виділяє необхідні ресурси для поточного сеансу, наприклад, з'єднання TCP / IP з тренажером. Сеанс повинен бути закритим, наприклад, при виході з програми шляхом розміщення ресурсів. Методи ініціалізації та розпорядження представлені наступним чином:

```
/* ініціалізує необхідний ресурс., з’єднання с симулятором*/
void SHAPI_Initialize();

/* звільніть ресурси наприклад., відключившись від симулятора */
void SHAPI_Dispose();
```

Запити на перерахування :

Динамічний макет та ієрархічний характер SmartHome вимагає методу перерахування різних сутностей. Наприклад, нам може знадобитися з’ясувати, скільки пристроїв знаходиться в конкретній кімнаті певного поверху. Також динамічне планування тренажера дозволяє змінювати кількість поверхів і кількість кімнат на кожному поверсі. Ми можемо використовувати наступний метод для виконання запитів перерахування :

```
/* отримати скільки поверхів в SmartHome   */
int SHAPI_GetFloorCount();

/* отримати скільки номерів у вказаному номері поверху  */
int SHAPI_GetRoomCount(int floorNumber);

/* отримати скільки пристроїв у даній кімнаті даного номера поверху */
int SHAPI_GetDeviceCount(int floorNumber, int roomNumber);

/* отримати кількість пристроїв у парі з пристроєм, визначеним id пристр. */
int SHAPI_GetPairedDevicesCount(int deviceId);
```

Пошук пристроїв :

Усі операції на пристроях виконуються за допомогою їх ідентифікатора пристрою. Ці ідентифікатори можна отримати за допомогою методів пошуку на пристрої. Результатом функцій є вектор ідентифікаторів

/* отримати id пристрою для всіх пристроїв у вказаному номері кімнаті та номеру поверху */

vector<int>* SHAPI_GetDevices(int floorNumber, int roomNumber);

/* отримати id всіх парних пристроїв против заданого id пристрою*/

vector<int>* SHAPI_GetPairedDevices(int deviceId);

Імена функцій :

Як вже згадувалося раніше, будівництво, підлоги, кімнати та пристрої - всі названі об'єкти. Їх імена можна отримати за допомогою наступних методів

/* отримати назву SmartHome дому , наприклад., Cool Mansion */

string SHAPI_GetBuildingName();

/* отримати ім’я поверху, наприклад., Second Floor */

string SHAPI_GetFloorName(int floorNumber);

/*отримати ім’я кімнати на певному поверсі, наприклад., Living Room */

string SHAPI_GetRoomName(int floorNumber, int roomNumber);

/* отримати ім’я пристрою из заданим id, наприклад., CD Player */

string SHAPI_GetDeviceName(int deviceId);

Атрибути читання та запису :

Читайте та записуйте різні атрибути сутностей у розумному будинку

/* отримати адресу SmartHome будинку */

string SHAPI_GetBuildingStreetAddress();

/* перевірка, чи пристрій увімкнено, чи вимкнено, повернене значення являє собою статус включеного*/

bool SHAPI_GetDevicePoweredOn(int deviceId);

/* перевірка, чи пристрій в Інтернеті чи офлайн, повернене значення відображає стан онлайн*/

bool SHAPI_GetDeviceOnlineStatus(int deviceId);

/* пристрій увімкнено або вимкнено, повернене значення представляє новий статус увімкнено*/

bool SHAPI_SetDevicePoweredOn(int deviceId, bool newStatus);

/* встановлює пристрій в режимі онлайн або в режимі офлайн, повернене значення представляє новий статус у режимі онлайн*/

bool SHAPI_SetDeviceOnlineStatus(int deviceId, bool newStatus);

Оцінюйте можливості пристрою :

Як вже було сказано раніше, різні пристрої мають додаткові можливості, які можна динамічно запитувати за допомогою наступних функцій

/* перевірка, чи вказаний пристрій повертає текстові повідомлення*/

bool SHAPI_IsTextEnabled(int deviceId);

/* перевірка, чи приймає даний пристрій команди*/

bool SHAPI_IsCommandEnabled(int deviceId);

/* перевірка, чи даний пристрій пов’язаний з безпекою */

bool SHAPI_IsSafetyRelated(int deviceId);

/* отримати поточний статус тексту із заданого текстового пристрою*/

```
string SHAPI_GetTextStatus(int deviceId);

/*отримати поточний стан безпеки даного пристрою, пов’язаного з безпекою*/
bool SHAPI_GetSafetyStatus(int deviceId);

/* отримати максимальний id команди з пристрою, що підтримує команду, min команда id завжди 0 */
int SHAPI_GetMaxCommandId(int deviceId);

/* виконати команду між 0 і максимальним id команди на даному пристрої з включеною командою*/
bool SHAPI_ExecuteCommand(int deviceId, int commandId);
```

Додаток Г

Агент системи моніторингу та управління «умним домом»

Схема роботи «HomeAssistant»

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ62141_20Б_14

Аркушів 1

Київ – 2020


Завантажити з файлу
Налаштування
Введення через UI
Система тестуваннння
Виконати тест
Зберегти у файл
Завантажити результати
Actor
Система "Home Assistant"
Авторизація
Планувальник дій
Переглянути список запланованих дій
Створити нову дію
Видалити дію
Редагувати дію
Моніторинг стану обладнання
Управління електропристроями
Виконати дію
Контроль температури
Контроль пожежної сигналізації
Контроль датчиків присутності
Контроль закриття дверей
Контроль sms сповіщень